# Системноинженерное мышление

Книга А.Левенчука “Системноинженерное мышление” — это вторая редакция учебника, первая редакция которого использовалась как в магистерских программах ВУЗов (семестровый курс 2014 года для пятикурсников межфакультетской кафедры технологического предпринимательства МФТИ в весеннем семестре 2014г., интенсивный курс магистратуры Высшей школы инжиниринга УрФУ в осеннем семестре 2014г.), так и в системах повышения квалификации инженеров и менеджеров (четырёхдневный тьюториал зимой-весной 2014г. для двух групп программы Global Professional корпоративной академии Росатома, проект повышения квалификации сотрудников ВНИИХОЛОДМАШа, проектная сессия НТЦ ФСК ЕЭС).

Вторая редакция книги (первая вышла под названием “Системноинженерное мышление в управлении жизненным циклом” в июне 2014г.) существенно дополнена разделом по используемым формализмам, появились ссылки на материалы по отдельным практикам системной инженерии. Но и в нашей второй редакции книги внимание уделено не гладкости и художественности изложения, не качеству вёрстки и иллюстраций, и даже не попыткам понятно изложить сложный материал. Главное было компактно собрать в одном тексте «мыслительный минимум» по системной инженерии, обычно рассыпанные по самым разным источникам знания. Специфика этого учебника в том, что его содержание базируется непосредственно на международных стандартах системной инженерии, разработанных или обновлённых за последние пять-шесть лет — ISO 15288, ISO 42010, ISO 15926, IEC 81346, OMG Essence.

Как объясняется в тексте, в нашей книге нет ни одного слова, которое встретится в жизни — а в жизни нет ни одного слова, которое есть в книге. После чтения книги предполагается групповой тренинг отождествления её материала и материала жизни — этот тренинг ведётся на “живых” проектах участников учебной группы. В ВУЗах это будут традиционные семинарские занятия, в компаниях проектные сессии: мы ожидаем flip teaching — “перевёрнутое обучение”, когда преподаватель/консультант не читает лекции и не объясняет новый материал, зато помогает выполнять упражнения — т.е. помогает в “домашних заданиях”. Мы также предполагаем, что “домашние задания” задаёт жизнь, они не берутся из книги. В

тексте книги мало примеров и упражнений, но это не означает, что учебник сугубо "теоретический". Просто нет планов делать нашу книгу самоучителем, добавлять чуждые задачи и "учебные кейсы" — книжные учебные примеры всё одно не пойдут впрок, они не будут волновать так, как волнуют собственные проекты.

Обычно живое обсуждение проектов с преподавателем/консультантом приводит к желанию повторно прочесть нашу книгу, в том числе заглядывая в дополнительную литературу, на которую дано много ссылок. Только и повторного прочтения обычно мало для полноценного освоения материала и умения применить его на практике.

Прорыв в понимании получается тогда, когда для освоения системноинженерного мышления каждый участник группы в обязательном порядке пишет эссе по приложению материала книги к своему рабочему проекту. Это заставляет по-настоящему продумать все разделы книги в их взаимосвязи между собой и с жизнью. Вот примерный план такого эссе:

1. Краткое описание проекта создания целевой системы, структурированное по всем основным альфам проекта, включая их важнейшие подальфы – их характеристика, текущее состояние, основные проблемы и рекомендации по их устранению. Особое внимание должно быть уделено вопросу как потребности фокусируют требования, которые фокусируют архитектуру, а потом как в соответствии с ними происходят проверка и приёмка.

2. Краткая характеристика жизненного цикла целевой системы, выбранного вида жизненного цикла проекта, практик управления жизненным циклом (управление конфигурацией, управление информацией и т.д.). Проблемы, рекомендации по их устранению. Особое внимание должно быть уделено вопросу насколько планы проектного управления и финансовые планы отражают архитектуру системы и особенности её жизненного цикла.

3. Примеры задействования в проекте практик системной инженерии (прежде всего – инженерии требований, инженерии системной архитектуры, проверки и приёмки, моделеориентированной системной инженерии), краткая характеристика используемых технологий, описание результатов их использования (что удалось улучшить).

4. Как устроено предпринятие по созданию системы, рекомендации по его улучшению.

5. Рефлексия: случилась ли метанойя и по каким именно темам, что осталось непонятным в материале курса, какие планы по продолжению образования.

Если курс двухсеместровый, то пункты 1-2 обычно пишутся в ходе первого семестра, в ходе второго семестра эти пункты обновляются и к ним дописывается содержание пунктов 3-5. Особое внимание уделяется тому, что в эссе не пересказывается и не цитируется материал книги, он только прикладывается к материалу целевого проекта.

Идеальный вариант, это когда текст эссе используется в отчётных материалах по рабочему проекту. Так решается проблема совмещения "фундаментального образования" (освоение материала нашей книги) и "практического образования" (выполнение конкретных рабочих проектов — производственных или учебных) — ибо плохо будет и с попытками выполнять проекты без теории и с попытками освоить теорию без выполнения проектов. Никакие задачи и упражнения не заменят проектной работы!

Книга даёт определения для требований, архитектуры, проверки и приёмки, других

понятий системной инженерии. Но книга не рассказывает о том, как разработать качественные требования и архитектуру, как тщательно провести проверку и приёмку системы, то есть книга не содержит описания практик современной моделеориентированной системной инженерии (хотя и содержит отсылки к соответствующей литературе). Изучение практик обычно требует дополнительных долгосрочных усилий, но этому изучению должно предшествовать знакомство с системноинженерным мышлением.

После освоения материала книги по системноинженерному мышлению продолжать образование системного инженера можно в двух противоположных направлениях:

- "дьявол в деталях": конкретизировать отдельные инженерные дисциплины, изучать практики моделеориентированной системной инженерии. Это традиционное обучение инженерии в её связи с реальной жизнью.
- "ангел в абстракциях" ("знание принципов освобождает от знания фактов"): обобщить предлагаемое системноинженерное мышление с целью достижения мультидисциплинарности и распространения его на самые разные виды систем. Обучать методологии инженерии во взаимосвязи с достижениями науки.

Раздел инженерии предпринятия факультативен и дан не столько для подробного изучения, сколько для ознакомления с тем, как изложенные принципы системноинженерного мышления применяются к нестандартным для классической инженерии сложным системам.

Изложение системноинженерного мышления, описанного в книге, совместимо с изложениями системного мышления в технологическом и инженерном менеджменте.

Раздел «Формализмы системной инженерии» был существенно отредактирован Виктором Агроскиным, им же в этот раздел добавлена математика и значительная часть примеров.

Активное участие в подготовке книги приняли преподаватели, аспиранты и студенты кафедры технологического предпринимательства Роснано. Без их активного участия вряд ли эта книга была бы написана.

Материалы книги неоднократно обсуждались на заседаниях Русских отделений INCOSE и SEMAT, автор выражает благодарность членам этих международных организаций за многочисленные замечания и предложения. Много ценных замечаний было представлено читателями блога автора (http://ailev.ru).

Ваши замечания и предложения по поводу книги присылайте Анатолию Левенчуку (ailev@asmp.msk.su).

Новости по книге будут появляться в блоге http://ailev.ru

Свежие версии книги будут появляться тут: http://techinvestlab.ru/systems_engineering_thinking/

# Оглавление

# 1. Системная инженерия

Системноинженерное мышление — это использование системного подхода в инженерии. Чтобы понять, где и зачем используется системноинженерное мышление, нужно сначала разобраться с тем, что такое системная инженерия и зачем она нужна.

## *Определения системной инженерии*

Самое современное определение системной инженерии дано в Guide to the Systems Engineering Body of Knowledge (руководство по корпусу знаний системной инженерии, http://www.sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_%28SEBoK%29, текущая версия 1.3.1 от 5 декабря 2014г.).

Короткое определение: системная инженерия — это междисциплинарный подход и способы обеспечения воплощения успешной системы (*Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems* — *http://sebokwiki.org/wiki/Systems_Engineering_%28glossary%29*). В этом определении можно подчеркнуть:

- Успешные системы — это то, чем занимается системная инженерия. Слово "успешные" тут крайне важно и означает, что система должна удовлетворить нужды заказчиков, пользователей и других стейкхолдеров (стейкхолдеры — это те, кто затрагивается системой, или кто затрагивает систему). Успех — это когда системой все довольны.
- Слово "системы" используется в очень специальном значении: это "системы" из системного подхода. Для системной инженерии слово "система" примерно то же, что "физическое тело" для ньютоновской механики — если вы сказали про компьютер "физическое тело", то это автоматически влечёт за собой разговор про массу, потенциальную энергию, модуль упругости, температуру и т.д. Если вы сказали "система" про компьютер, то это автоматически влечёт за собой разговор про стейкхолдеров и их интересы, требования и архитектуру, жизненный цикл и т.д.
- Междисциплинарный подход — системная инженерия претендует на то, что она работает со всеми остальными инженерными специальностями (впрочем, не только инженерными). "Подход" обычно означает какие-то наработки в одной предметной области, которые можно перенести на другие предметные области. Междисциплинарность — это очень сильное заявление, оно означает, что системная инженерия может в одну упряжку впрячь коня и трепетную лань (например, инженеров-механиков, баллистиков, криогенщиков, психологов, медиков, астрономов, программистов и т.д. в проектах пилотируемой космонавтики).
- Слово "воплощение" (realization, "перевод в реальность") означает буквально это: создание материальной (из вещества и полей) успешной системы.

По-английски "системная инженерия" — systems engineering, хотя более ранние написания были как system engineering. Правильная интерпретация (и правильный перевод) — именно "системная" (подразумевающая использование системного подхода) инженерия, а не "инженерия систем" (engineering of systems) — когда любой "объект" обзывается "системой", но не используется системный подход во всей его полноте.

Дискуссия о значении термина в переводе "с английского на английский" тоже существует, поэтому есть чёткие сформулированные ответы на эту тему. В этой дискуссии про "systems engineering vs. engineering of systems" сами системные инженеры (в отличие от разных других инженеров — механиков, электриков, программистов и т.д.) заняли чёткую и согласованную коллективную позицию. Она, например, выражена в материале профессора Derek Hitchins "Systems Engineering vs. Engineering of Systems – Semantics?": http://www.hitchins.net/profs-stuff/profs-blog/systems-engineering-vs.html.

Почитайте, в тексте по ссылке рассказана очень любопытная история, включая экскурс в английскую грамматику для данного случая. Грубо говоря, под "инженерией систем" (например, control systems engineering, manufacturing systems engineering) понимается что ни попадя: легко выкинуть слово "система", которое лишь обозначает некий "научный лоск". Инженеры легко любой объект называют

"системой", не задумываясь об осознанном использовании при этом системного мышления, не используя системный подход, "инженерия управляющей системы" это просто "инженерия управляющего объекта". Никакого специального мышления при этом не предусматривается, слова "система" и "объект" взаимозаменяемы. В самом лучшем случае про систему скажут, что "она состоит из взаимодействующих частей" — на этом обычно разговор про "систему" и "системность" заканчивается, он не длится больше двадцати секунд.

А вот из системной инженерии квалификатор "системный" без изменения смысла понятия выкинуть нельзя — системная инженерия это инженерия с системным мышлением в голове (а не любая инженерия, занимающаяся объектами, торжественно поименованными системами просто для "добавления сложности и научности").

Поэтому желающие заниматься "инженерией систем" — не трогайте их, пусть делают своё частное узкое специальное инженерное дело, они очень полезны и нужны, но они не системные инженеры.

В последние годы увеличилось количество переводов инженерной литературы, и слово engineering не удосуживаются перевести как "инженерия", так и оставляют "инжинирингом". Перевод "системный инжиниринг" уже начинает побеждать — это легко отследить по результатам сравнения в интернет-поисковых системах. Можно считать, что "системная инженерия" и "системный инжиниринг" синонимы, но есть маленькая проблема: в России почему-то в тех местах, где занимаются инженерным менеджментом, а не инженерией, называют его "системным инжинирингом". Так что будем считать "инженерию" и "инжиниринг" синонимами, но в случае "инжиниринга" проверять на всякий случай, не менеджмент ли имеется ввиду вместо чисто инженерной работы.

Более длинное определение включает ещё одну фразу: "Она фокусируется на целостном и одновременном/параллельном понимании нужд стейкхолдеров; исследовании возможностей; документировании требований; и синтезировании, проверке, приёмке и постепенном появлении инженерных решений, в то время как в расчёт принимается полная проблема, от исследования концепции системы до вывода системы из эксплуатации" (*It focuses on holistically and concurrently understanding stakeholder needs; exploring opportunities; documenting requirements; and synthesizing, verifying, validating, and evolving solutions while considering the complete problem, from system concept exploration through system disposal).* Тут нужно подчеркнуть:

- Целокупность, которая подчёркнута многократно — от "междисциплинарности" в первой половине определения до целостности всех действий по созданию системы во второй половине определения, до целостности/полноты проблемы, до охвата всего жизненного цикла системы "от рождения до смерти". Целостность (полнота охвата всех частей целевой системы согласованным их целым), междисциплинарность (полнота охвата всех дисциплин) — это ключевое, что отличает системную инженерию от всех остальных инженерных дисциплин.
- Параллельность выполнения самых разных практик (а не последовательное выполнение их во времени, как можно было бы подумать, прочитав перечисление практик)
- Много особенностей, которые будут понятны позднее (различение нужд пользователей и требований, проверки и приёмки, упор на синтез для

противопоставления “аналитическим” дисциплинам и т.д.).

*Упражнение:* посмотрите разные определения системной инженерии и найдите их сходство и различия (http://www.sebokwiki.org/wiki/Systems_Engineering_%28glossary%29, http://www.sie.arizona.edu/sysengr/whatis/whatis.html, http://syse.pdx.edu/program/about.php и так далее — погуглите, чтобы найти больше определений). Обратите внимание на разнообразие используемой в этих определениях терминологии.

### Ответственность за целокупность и междисциплинарность

Ответственность за всю систему как целое (whole system) и связанная с этим межпредметность/междисциплинарность (interdisciplinary) подхода к другим инженериям (механической, электрической, программной, предприятия и т.д.) отличают системную инженерию от всех других инженерных дисциплин. Представим себе ледовую буровую платформу:

Сотни тысяч тонн металла, бетона, пластмассы, необходимых расходных материалов, обученная вахта должны собраться вместе далеко в море среди льда и в строго определённый момент эта огромная конструкция должна начать согласованно работать — и не просто работать, а приносить прибыль и обеспечивать безопасность в части загрязнения окружающей среды и здоровья находящейся на платформе вахты. Какая инженерная дисциплина должна учесть результаты работ всех других инженерных дисциплин — собрать в единое целое данные ледовой обстановки, санитарных норм в помещениях для обслуживающего персонала, обеспечение электричеством попавших туда компьютерных серверов, характеристики этих серверов и программное обеспечение? Кто озаботится учётом в конструкции платформы изменений в длине металлоконструкций за счёт разницы суточных температур и одновременно установкой акустических датчиков на трубах, которые прослушивают шорох песка, чтобы по этому шороху можно было определить износ труб?

Системная инженерия как раз и является той дисциплиной, которая ответственна за обеспечение целостности в инженерном проекте: именно системные инженеры проектируют нефтяную платформу как успешное (безопасное, надёжное,

прибыльное, ремонтопригодное и т.д. — разным людям нужно от этой платформы разное) целое, потом раздают части работы инженерам по специальностям (инженерам-строителям, машиностроителям, инженерам-электрикам, компьютерщикам/айтишникам и т.д.), а затем собирают результаты их работ так, чтобы получить работоспособную и надёжную систему.

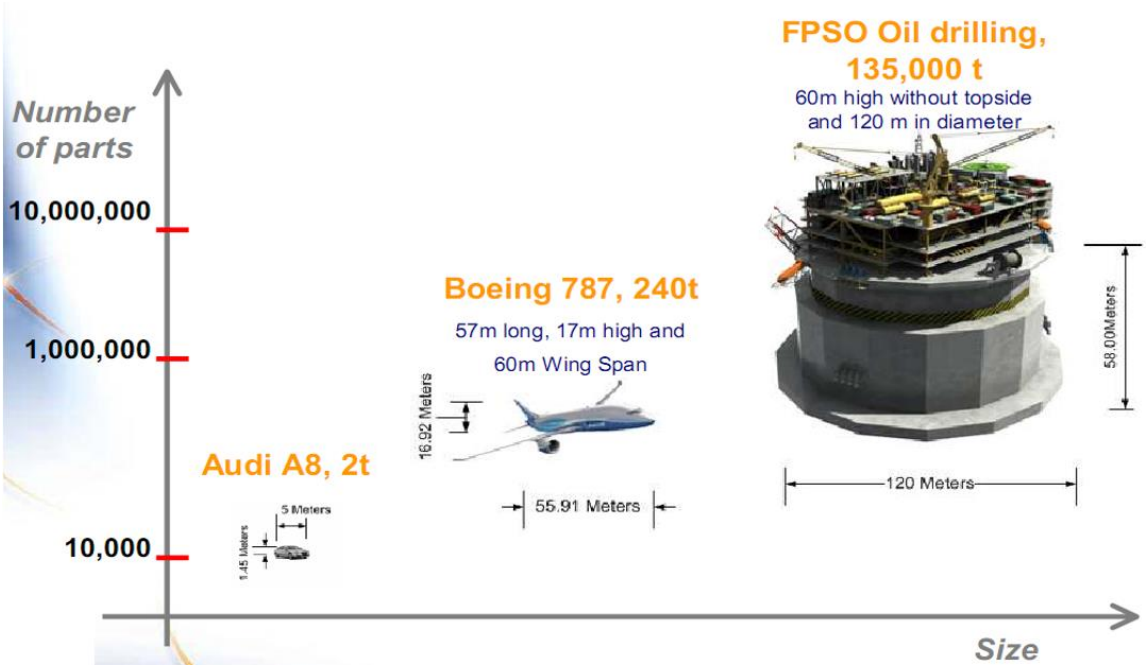
Это главный признак, отличающий системных инженеров от всех других инженеров: они отвечают за проект в целом, сразу во всех его деталях как в части деталей-частей, так и в части использования детальных знаний отдельных дисциплин. Они ответственны за то, чтобы не было пропущено какой-нибудь мелочи, ведущей к провалу.

Самолёт — это много-много кусков металла и пластика, синхронно летящих на скорости 900км/час (0.85 от скорости звука, это типовая скорость Boeing 787 Dreamliner) на высоте 10км. Системный инженер — это тот, кто придумал, как обеспечить их надёжный и экономичный совместный полёт, увязав самые разные требования (грузоподъёмность, расход топлива, дальность полёта, шум при взлёте и посадке, требования к длине разбега и посадки, необходимость лёгкого обслуживания на земле, отсутствие обледенения, безопасность людей на борту, и т.д. и т.п.), при этом требования выдвигались самыми разными людьми, представляющими самые разные профессиональные и общественные группы. Пара-тройка миллионов деталей изготавливается и собирается в одно изделие — и самолёт летит, обеспечивая комфорт пассажирам и прибыль владельцам!

### Для чего нужна системная инженерия: победить сложность

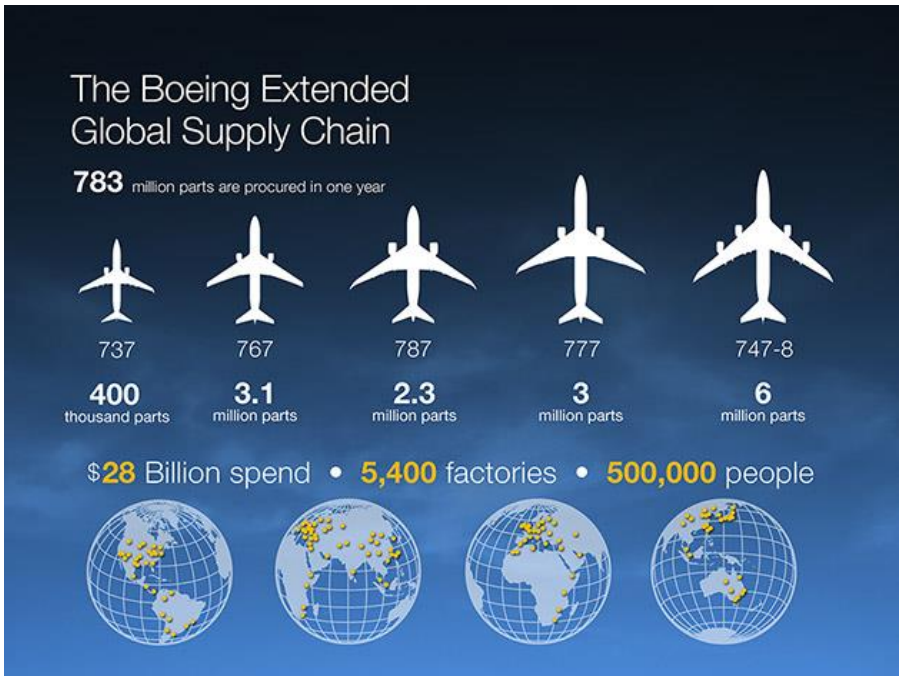
Системная инженерия решает задачу преодоления инженерной сложности (которая определяется главным образом как число различных элементов, которые должны провзаимодействовать друг с другом, чтобы получить целевую систему — и сложные системы не помещаются ни в какую самую умную голову, требуется специальная дисциплина, чтобы собирать результаты деятельности самых разных инженеров в одно работоспособное целое).

Микроволновка содержала в 2006 году 212 частей (http://www.cadenas.de/news/en/reader/items/out-of-many-one-a-surprisingly-large-number-of-parts), но если взять системы посложней, то счёт легко переходит на миллионы частей — и во многих крупных системах ошибки в любой из частей могут приводить к катастрофам.




(картинка из материалов Dassault Systemes)

На самом деле картинка эта приукрашивает ситуацию со сложностью. Вот более точная информация по самолётам Boeing (http://787updates.newairplane.com/787-Suppliers/World-Class-Supplier-Quality):




В современных системах число отдельных элементов, которые нужно согласовать между собой (в проектировании), а часто и создать с нуля (в конструировании) достигает десятков миллионов в “железных” системах (а на одном серийно выпускаемом электронном чипе FPGA Xilinx Virtex-Ultrascale XCVU440 число отдельных транзисторов оценивается на 2014 год более чем в 20 миллиардов — http://en.wikipedia.org/wiki/Transistor_count).

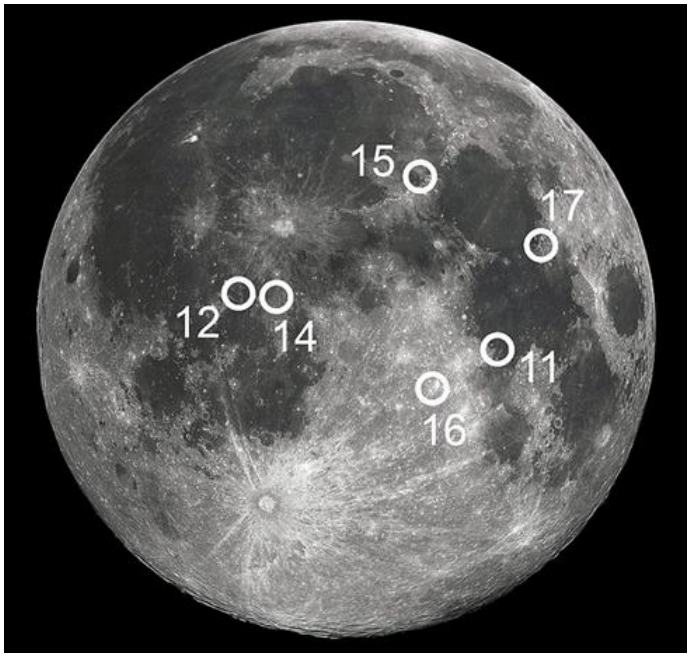
Согласно легенде, системная инженерия впервые появилась как метод ведения работ в военной отрасли США, когда нужно было скрестить два сверхсложных инженерных проекта: атомный проект по созданию ядерного оружия и проект создания баллистических ракет, необходимых для доставки этого оружия. Не было никаких голов “генеральных конструкторов”, которые могли были бы справиться с

решением этой задачи, и пришлось изобретать методы совладания со сложностью подобного сверхпроекта.

Системная инженерия делает невозможное возможным: прежде всего это относится к тому, что сверхсложные системы работают так, как задумано. Системные инженеры больше всего работают на этапах, когда самой системы ещё нет, и их задача представить правильный проект, в котором разные части системы разные свойства, которыми занимаются разные инженерные дисциплины, влияют друг на друга так, чтобы не было неожиданностей от этих влияний. Системный инженер ответственен за то, чтобы риска неудачи, связанного с плохой конструкцией, не было. Системный инженер ответственен за то, чтобы невозможно сложные системы за невозможно короткое время (это означает прежде всего отказ от тупого метода проб и ошибок, тем более что в сверхсложных системах нельзя сделать всех возможных "проб", чтобы получить все возможные ошибки) создавались, и работали как от них ожидается.

Раньше люди боролись со сложностью тем, что инженерными проектами занимались гении. Сейчас на гениальность не надеются: проекты должны получаться успешными независимо от гениальности "генерального конструктора", они должны получаться успешными на основе метода — гениальность при этом позволяет дополнительно поднимать планку сложности, уже высоко поднятую использованием системноинженерного мышления и практик системной инженерии.

Классический пример, показывающий работоспособность системной инженерии — это "аэрокосмос" (aerospace), авиационная и космическая промышленности. Как известно, обеспечить надёжные космические полёты очень трудно, до сих пор. Много лет бытует поговорка про самые трудные дела — что это "rocket science". Именно в "аэрокосмосе" без использования системной инженерии не удавалось добиться успехов. Основа конструирования космических кораблей в NASA и европейском космическом агентстве — это системная инженерия. Ещё в 1969-1972 годах в рамках программы Apollo (вдумайтесь: это более 40 лет назад!) на лунной орбите побывало 24 человека (из них трое — дважды!), а на Луне гуляло 12 космонавтов, они привезли на Землю 382 кг лунного грунта (http://en.wikipedia.org/wiki/Apollo_program).



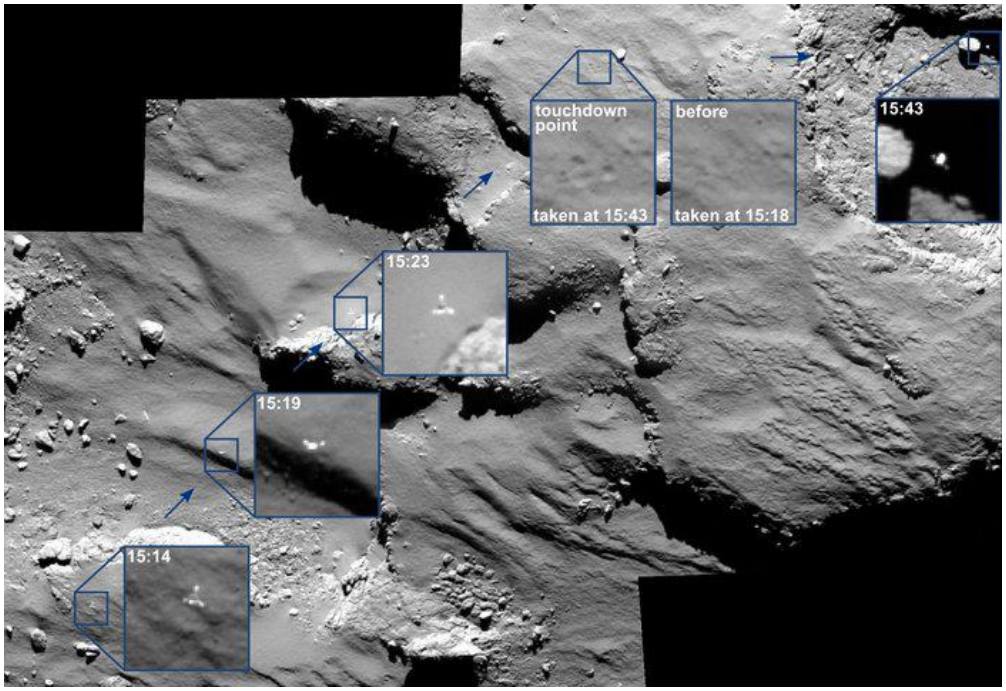

(места посадки миссий Apollo).

В программе Apollo использовались двухместные автомобили-планетоходы аж в трёх экспедициях из шести, и в экспедиции Apollo-16 даже был поставлен рекорд скорости передвижения по Луне на автомобиле: 18 км/час (такая скорость на Луне огромна: ведь там сила тяжести вшестеро меньше, и дороги отнюдь не асфальтовые, так что автомобиль на такой скорости ощутимо подбрасывало). На автомобиле американцы проехали почти 36км только в экспедиции Appolo-17, удаляясь от лунного модуля на расстояние до 7.6км.

Марсоход Opportunity прошёл за десять лет службы более 40 км, но уже не по Луне, а по Марсу — с 2004г. (http://en.wikipedia.org/wiki/Apollo_Lunar_Roving_Vehicle, там выразительная картинка).

Но это всё достижения далёкого уже прошлого. Нынешние инженеры ставят принципиально другие задачи. В 2014 году Индия стала страной, чей космический зонд достиг орбиты Марса с первой попытки (кстати, это один из лозунгов системной инженерии: "С первого раза правильно!" — и тут нужно учесть, что из 40 полётов на Марс меньше половины оказались успешными) раза, и при этом запуск обошёлся всего в US$73 млн., самый дешёвый полёт на Марс на сегодня (http://en.wikipedia.org/wiki/Mars_Orbiter_Mission). Кто это делал? Системные инженеры. Например системный инженер Minal Sampath возглавляла команду, создавшую три прибора для этого марсианского спутника (http://www.bbc.com/news/world-asia-25989262):


Системные инженеры из европейского космического агентства в 2014 году успешно выполнили миссию Rosetta — посадили спускаемый аппарат Philae с 12 измерительными приборами на комету диаметром примерно 4км. Чтобы попасть точно в назначенную точку и в точно назначенное время Rosetta стартовала с Земли в 2004г., а в 2012 Rosetta удалялась от Земли на миллиард километров (и на 0.8млрд. Километров от Солнца) — http://www.esa.int/Our_Activities/Space_Science/Rosetta/Frequently_asked_questions. В момент встречи с кометой Rosetta была на расстоянии 405млн.км и они летели по направлению к Солнцу со скоростью примерно 55тыс.км/час (http://www.esa.int/Our_Activities/Space_Science/Rosetta/Rosetta_arrives_at_comet_destination). Вот фотография спуска модуля Philae, сделанная камерой Rosetta (в этот момент Rosetta была на высоте примерно 15.5км над поверхностью кометы, выделенные квадраты имеют сторону 17м — http://www.esa.int/spaceinimages/Images/2014/11/OSIRIS_spots_Philae_drifting_across_the_comet):




Основатель космической фирмы SpaceX (http://www.spacex.com/) Элон Маск поставил цель снизить стоимость запуска в 10 раз за счёт повторного использования первой ступени ракеты — чтобы затраты на запуск приходились в основном на топливо, как сегодня у самолётов. Посадочная площадка у него — платформа, которая плавает в море: http://www.spacex.com/news/2014/12/16/x-marks-spot-falcon-9-attempts-ocean-platform-landing. В 2015 году Элон Маск обещал раскрыть планы по подготовке его фирмы к колонизации Марса. Какой метод работы? Такой же, как в NASA — системная инженерия.

Фирма Virgin Galactic (http://en.wikipedia.org/wiki/Virgin_Galactic) планирует осуществлять через год-два относительно недорогие экскурсии в космос (без выхода на орбиту, но по баллистической траектории с шестью минутами невесомости), за год после начала деятельности планируя удвоить число космонавтов (поднимавшихся на высоту более 100км над поверхностью Земли) — на 8 июня 2013г. на Земле было 532 космонавта из 36 стран.

Подводная лодка класса Astute (Великобритания) имеет более миллиона комплектующих (http://gregornot.wordpress.com/2008/10/22/one-of-the-most-complex-engineering-projects-in-world%E2%80%99/), атомные электростанции имеют порядка 4 млн. комплектующих (только число их крупных подсистем ГИПы оценивают в 400-700), а гражданские пассаржирские суда класса Oasis лишь немного уступают авианосцам в их размерах (http://en.wikipedia.org/wiki/Oasis_class_cruise_ship).

Поезд со скоростью 200км в час чудо только в России (рекорд скорости железной дороги 210км/час был достигнут ещё в октябре 1903г. — http://en.wikipedia.org/wiki/High-speed_rail), но в 2008 Китай открыл линию "Wuhan – Guangzhou", штатная скорость на которой до июля 2011 года былат 350 км/час, после чего была снижена до всего 300км/час. Начата системноинженерная работа по проекту Hyperloop, где вагончики будут двигаться со скоростями, близкими к 1000км/ч в трубах с пониженным давлением воздуха (http://en.wikipedia.org/wiki/Hyperloop).

Небоскрёбы уже строят высотой более 1км (Kingdom Tower, Jeddah Saudi Arabia, начало строительства в 2013 году, окончание в 2019, 167 этажей, высота 1007

метров, http://skyscraperpage.com/diagrams/?searchID=202). Уже построено здание высотой 828 метров (Burj Khalifa, Dubai Unated Arab Emirates, построено в 2010 году, http://skyscraperpage.com/diagrams/?searchID=200).

Кто ответственен за правильность проекта, правильность выбранного решения, соблюдение всех сложнейших практик, дающих на выходе "правильно с первого раза"? Системные инженеры: именно они должны придумать такие технические решения, чтобы не было неприятных (впрочем, и приятных тоже) неожиданностей при изготовлении, эксплуатации и последующем выводе из эксплуатации систем, создаваемых во всех этих сложнейших проектах. И эти системные инженеры справляются — каждый год сверхсложные проекты прошлых лет становятся просто сложными, а сложные проекты — типовыми. Это не значит, что такие проекты, как полёт на луну или создание атомной станции становятся более простыми. Нет, не становятся: наоборот. Сложность этих проектов год от года растёт за счёт опережающего роста требований, прежде всего требований безопасности.

Создавать такие сложные системы могут только большие многодисциплинарные коллективы, которые требуют какой-то междисциплинарной организации в разделении умственного труда. Именно вопросы удержания междисциплинарной целостности и организации междисциплинарных работ и решает системная инженерия. Она:

- удерживает целое всего инженерного решения для самых разных затрагиваемых инженерным проектом людей (стейкхолдеров). Это основной предмет нашей книги.
- Использует практики системной инженерии для создания успешного инженерного решения — наша книга не рассказывает подробно об этих практиках, но даёт ссылки на дополнительную литературу.

Системная инженерия стремительно развивается. В июне 2014 года международный совет по системной инженерии (INCOSE) выпустил INCOSE Systems Engineering Vision 2025 – «вИдение» того, какой системная инженерия будет через десять лет, в 2025 году: http://www.incose.org/newsevents/announcements/docs/SystemsEngineeringVision_2025_June2014.pdf

Можно ли системную инженерию применять в простых проектах? Да, "кашу маслом не испортишь", хорошее мышление и хорошие инструменты помогают везде. В том числе для небольших проектов использование системной инженерии снижает требования к гениальности инженеров: думать помогает метод (и компьютеры — в силу понимания, что они должны делать), а не "общая гениальность". Нужно ли "просто инженерам" быть знакомым с системной инженерией? Да, они будут лучше понимать, как им взаимодействовать с многочисленными стейкхолдерами и как обеспечить успешность своей системы. Впрочем, "просто инженеров" часто уже учат основным практикам системной инженерии, хотя и неявно и без опоры на системное мышление. Нужно ли быть знакомым с системноинженерным мышлением менеджерам? Да, если они хотят понимать самых разных инженеров и сотрудничать с ними.

Но почему системную инженерию назвали именно системной, а не какой-то другой инженерией? Потому как на сегодняшний день единственным способом удержать сверхсложное целое в междисциплинарных проектах является использование системного подхода, в котором термин "система" используется в специальном смысле, и который подразумевает специальное устройство мышления для

применяющих системный подход людей. Этому мышлению и посвящена наша книга.

*Упражнение:* оцените, сколько отдельных частей в разрабатываемой в вашем проекте системе. Слышали ли вы, что в проекте явно используется какой-то метод работы (необязательно “системная инженерия”, но хоть какой-то, о котором написан учебник, который преподаётся в ВУЗе)? Как вы считаете, почему при всех разговорах о лидерстве советской космонавтики ей не удалось при практически неограниченных ресурсах послать космонавтов на Луну в начале 70-х?

## *Профессия системного инженера*

### Системный инженер как профессия

Системный инженер — это тот, кто отвечает за успешность системы в целом. Те, кто занимаются системной инженерией, называются системными инженерами (systems engineer), а сама системная инженерия тем самым является профессией в классическом смысле этого слова: есть профессиональные ассоциации системных инженеров, проводятся профессиональные конференции, есть учебные курсы в системе высшего образования.

О профессии системного инженера хорошо рассказывается в паре видеолекций главного инженера NASA по миссиям в солнечной системе Gentry Lee: http://spacese.spacegrant.org/index.php?page=videos . Gentry Lee пытается показать, каким должен быть "идеальный системный инженер" и приводит много примеров из жизни NASA.

Системные инженеры должны быть техническими лидерами в инженерных коллективах. Понятие технического лидерства (technical leadership) означает помощь в организации коллективной мыслительной работы по отношению к той или иной технической идее: все участники проекта должны делать одну и ту же систему, а не разные. Отличия технических лидеров от “технических евангелистов”: евангелисты — это проповедники чужих идей на предмет их воплощения в самых разных проектах инженерной компании или даже отрасли, а системные инженеры как технические лидеры — сами себе “технические иисусы христы” в отдельных конкретных проектах, они сами поставщики тех технических решений, в которых потом они должны уметь убедить других людей в их конкретном проекте. Системные инженеры не просто берут идеи от одних инженеров, а потом убеждают других инженеров их принять. Системные инженеры генерируют технические идеи самостоятельно.

В статье "Наука и искусство системной инженерии" — http://www.worldscinet.com/srf/03/0302/free-access/S1793966609000080.pdf , отражающей опыт системной инженерии NASA приводится сравнение системного инженера с дирижёром — симфонический оркестр, который творит Симфонию под управлением специально обученного и талантливого дирижёра. Системный инженер, как дирижёр, налаживает работу "симфонического оркестра" из многих инженеров-по-специальности. Это и правда и неправда одновременно. Правда в том, что системные инженеры — это технические лидеры. Неправда в том, что системный инженер это один человек, “рулящий” огромными коллективами других инженеров. Как в инженерии произошло разделение на инженеров по специальности (механиков, электриков, программистов, теплотехников и т.д.), как в западной медицине произошло разделение врачей по разным врачебным специальностям, и врачи редко работают поодиночке, так подобное разделение уже произошло и в самой системной инженерии. Системных инженеров разной

специализации может быть в одном проекте целая команда.

Вспомните ситуацию начала времён WWW, когда появилась и начала бурно развиваться веб-мастеринг и профессия веб-мастера. Трудно уже вспомнить, но всего десять лет назад был один человек, который занимался для вебсайтов всем: программировал движок, разрабатывал арт-дизайн, пришивал его к движку, наполнял вебсайт материалом, продвигал его в Сети и т.д. Один человек, который совмещал в себе всё разнообразие specialty engineering для сайтостроения. Сейчас "вебмастера" уже нет, а есть отдельно программисты CMS, дизайнеры, верстальщики, редакторы (в вариантах editor и content manager), администраторы, модераторы, "информационные архитекторы" (не могу удержаться, чтобы не писать их в кавычках), SEO и это еще не полный список. При развитии профессии она дробится на различные профессиональные позиции. Один человек, даже если он гений, не в состоянии удержать целостность: целостность удерживается в современном мире только командно. Есть ли "системный инженер" в сайтостроительстве? Может быть, а может и не быть. Есть ли "системная инженерия"? Безусловно, есть — и работа с требованиями, и создание сайтовой архитектуры, и стыковка всего этого с работами многих specialty engineers. Не все работы "системных инженеров" выполняются одним человеком. В системной инженерии тоже есть специализации.

К профессии “системный инженер” нужно относиться примерно так же, как к профессии “врач”: с одной стороны, вы никогда не спутаете стоматолога и гинеколога, но с другой стороны, эти обе профессии врачебные. Системные инженеры так же разнообразны, как и врачи: системный инженер, занимающийся ракетами и системный инженер, занимающийся небоскрёбами, будут иметь как много различий, так и много общего. Но кроме различия в видах целевых систем, которые разрабатывают эти системные инженеры, есть и различия в том, что системные инженеры делают по отношению к проекту — как врачи в операционной делятся на хирургов и анестезиологов, так и системные инженеры могут быть инженерами по требованиям, инженерами-архитекторами, инженерами по тестированию/испытаниям. Инженерный проект, как и хирургическая операция требует наличия многих людей в операционной — и ещё бОльшего количества людей за пределами операционной (клинических лабораторий, терапевтов, поставщиков оборудования, фармакологов и т.д.).

Это сравнение системного инженера с дирижёром можно прокритиковать и с другой стороны (кстати, обратите внимание на книгу по самоорганизующимся системам "Оркестр играет без дирижёра", http://urss.ru/cgi-bin/db.pl?lang=Ru&blang=ru&page=Book&id=45524&list=42). Метафора симфонического оркестра соответствует административной модели управления, "писанной музыке", и чуть ли не "руководству" — буквально, “руками водству” (может возникнуть впечатление, что системный инженер непосредственно командует каждым инженером по специальности, когда и какую ему ноту играть, как это происходит иногда в симфонических оркестрах с авторитарным дирижёром). Но ведь ещё в прошлом, XX веке музыка по факту пошла другими путями:

- в симфонической музыке за счёт компьютерной поддержки композитора оказались выкинутыми и дирижёр, и его оркестр, а "симфонические" фонограммы к современным фильмам и играм готовятся самим композитором буквально в одиночку — но на компьютере;
- в музыкальной мейнстримной культуре преобладает джаз, подразумевающий совсем другие принципы коллективного творчества: импровизация плюс

взаимоподстраивание (рок — это тот же джаз, ибо в рок-группах никакого "дирижёра", а вместо "нот" используется звукозапись);

- симфонические оркестры остались как очень дорогое средство антикварного хранения традиции, и от них никакого развития музыки не ожидается. Развитие музыки и музицирования идёт, при этом идёт достаточно бурно, но в совершенно других формах.

Если посмотреть на то, что происходит в менеджменте и инженерии, то тренд к "джазовой" организации деятельности несомненен. Все движение agile ("гибкости", непредзаданности последовательности шагов) — это именно в ту сторону, и все остальные примеры новинок в организационных дисциплинах (например, переход от акцента на administration/management к leadership) именно в эту сторону отсутствия "единоличного лица, принимающего все ответственные решения". Ситуация, при которой все главные решения принимаются одним лицом, которое всеми "дирижирует", опасна. "Дирижёр всего" потенциально создаёт угрозу появления в проекте "бутылочного горлышка", существенно замедляющего принятие инженерных решений (ибо решений много, дирижёр один, все решения он не то что скоординировать — он просто познакомиться с ними не успеет толком)! Но главное тут даже не в замедлении работы: одному человеку иметь образование и опыт во всех дисциплинах, в которых принимаются важнейшие решения по проекту невозможно: гений в одних вопросах вполне закономерно может быть полным идиотом в других вопросах. Метафора "великого вождя" в системной инженерии не соответствует духу времени.

Системные инженеры специализируются до инженеров по требованиям, инженеров-архитекторов, инженеров по испытаниям, инженеров по безопасности и т.д., и далее целая команда таких людей не столько командует остальными инженерами (как по факту командует дирижёр музыкантами оркестра, сам ни на чём не играя), сколько организует их взаимодействие, выполняя свой кусок инженерной работы.

Вместо "дирижёрской" лучше использовать "джазовую" метафору описания деятельности. Так, в джазе есть звукооператор (producer, хотя это слово в России значит совсем другое) — он из записанных разными музыкантами отдельных треков делает окончательную запись. Но он не предписывает, какую музыку играть музыкантам. Или руководитель джазового ансамбля, который выбирает, какую мелодию будут играть — но он не командует кому, когда и что играть, и не выдаёт точные ноты. Системные инженеры в команде тоже имеют различающиеся функции. Но как музыкантов из ансамбля называют "музыкант", так и мы системных инженеров из команды будем называть "системный инженер".

Разные виды системных инженеров имеют и разные акценты в их образовании. Так, инженер по требованиям должен общаться с разными стейкхолдерами и иметь опыт улаживания конфликтов. Это означает, что в его подготовке должен быть дополнительный коммуникационный тренинг и курс конфликтологии. Инженер-архитектор должен владеть какими-то методологиями перевода проблем (которые непонятно, как решать) в последовательность задач (которые понятно как решать), например, методологией ТРИЗ+.

Главным же критерием отнесения какой-то инженерной специальности к системной инженерии является то, что системный инженер думает о всей системе в целом, а не о каком-то её аспекте (механическом, электрическом, программном и т.д.). Именно этот критерий даёт основание David Firesmith относить инженеров по

безопасности к системным инженерам: несмотря на то, что инженеры по безопасности имеют свои ВУЗы, профессиональные организации и конференции, они думают обо всей системе в целом, и на этом основании их вполне можно считать системными инженерами.

Конечно, при этом мы считаем, что речь идёт о системной инженерии (в головах у системных инженеров системный подход), а не о других видах инженерии, "инженерии систем" (в головах представителей всех других инженерных профессий совершенно необязательно используется системный подход, хотя они и могут называть любой объект "системой". Но "не всяк мужик, кто в штанах ходит").

Ещё одна классификация разных видов системных инженеров была предложена в 2013 году техническим директором INCOSE Bill Miller. Он предложил отнести всех системных инженеров к шести "племенам" (tribes) в соответствии с их основным интересом в системной инженерии:

1. Технические практики всего жизненного цикла (работа с требованиями, архитектурой, проверкой и приёмкой, и т.д.)
2. Системноинженерный менеджмент, который тоже озабочен практиками, только практики о другом (управление конфигурацией, управление информацией, и т.д.).
3. Моделеориентированная системная инженерия, которая пытается трансформировать практику использования неявных (и часто запертых в сером веществе инженеров) описательных и вторичных по отношению к спецификациям моделей в практику использования явных первичных и богатых выразительными возможностями моделей.
4. Нетрадиционные промышленные экосистемы (то есть за пределами аэрокосмической инженерии), часто включают и "местные" практики.
5. "Мягкие системы", системное мышление и системная наука, которые имеют дело со сложными, вероятностными или недетерминистскими системами.
6. Системноинженерое лидерство, заинтересовано объединить практики всех остальных племён и катализировать их дружную совместную работу.

"Системный инженер" (дальше мы будем говорить "системный инженер" так же, как мы говорим "врач" или "музыкант" — игнорируя тот факт, что речь идёт о множестве специализаций внутри профессии и командной работе этих специализаций в большинстве случаев, а также игнорируя деление системных инженеров на "племена") была признана в 2009 году лучшей профессией в США. В опубликованном CNN Money совместно с PayScale рейтинге работ с самой большой оплатой и карьерным ростом первое место занимает системный инженер (http://www.incose.org/newsevents/senews/docs/CNNMoney.pdf), второе место — ассистент врача, третье — профессор в колледже, четвертое — медсестра, пятое — менеджер проектов по информационным технологиям, шестое — сертифицированный бухгалтер. В 2009г. В США было 88 тыс. системных инженеров. Мнение рейтинга, почему эта работа так хороша:

> *Спрос на системных инженеров взлетает по мере того, как нишевая работа в аэрокосмической и оборонной промышленности становится привычным явлением среди разнообразного и расширяющегося разнообразия работников от разработчиков медицинского оборудования до корпораций типа Xerox и BMW.*

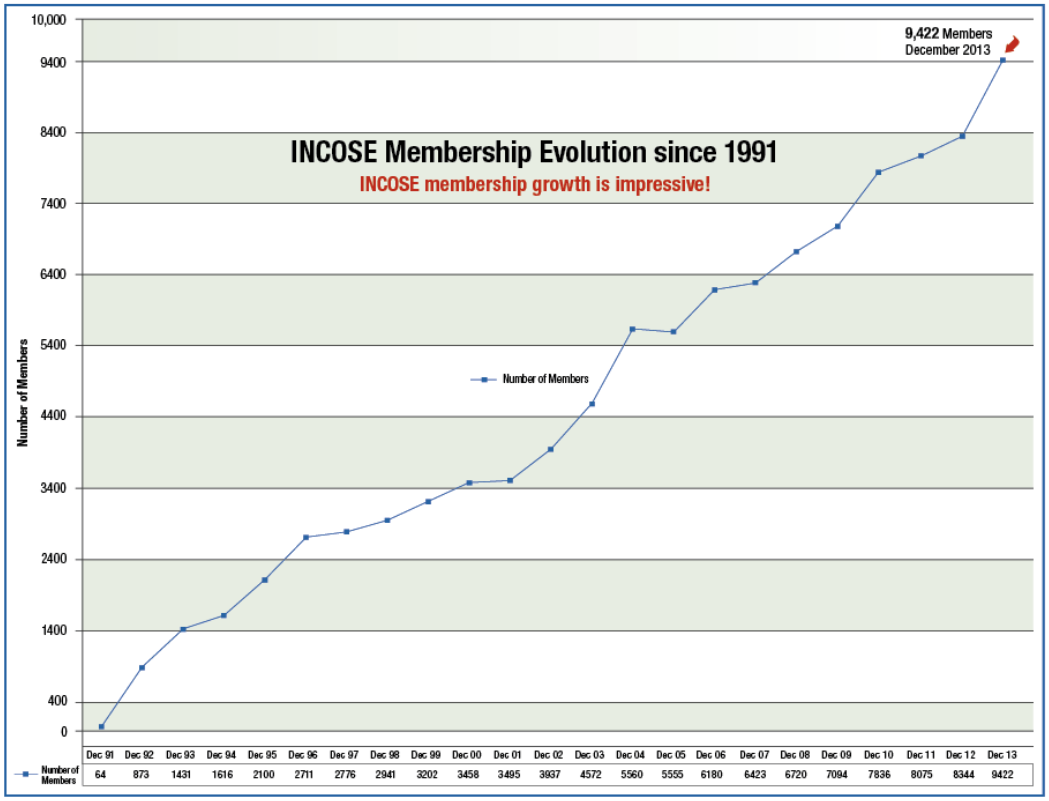
*Оплата может легко достигать шестизначных цифр для самых лучших, и существует достаточно возможностей для продвижения. Но многие системные инженеры говорят, что больше всего удовольствия им приносят творческие аспекты работы, а также зрелище осуществляющихся проектов. "Транспортная система, над которой я работаю, на самом деле делает значимые изменения для людей", говорт Anne O'Neil, главный системный инженер Нью-Йоркского транспортного агентства.*

Профессиональные организации системных инженеров

В мире существует несколько профессиональных организаций системных инженеров, из них наиболее значительной является международный совет по системной инженерии (INCOSE, International Counsil of Systems Engineering, http://incose.org).


В этой организации есть индивидуальное членство ($145 в год) и корпоративное членство (порядка $10 тыс. в год), индивидуальных членов примерно 10тыс. человек со всех стран мира.

Цели INCOSE — распространять знания по системной инженерии, обеспечивать международное сотрудничество и обмен опытом системных инженеров, устанавливать стандарты профессионального мастерства и проводить сертификацию системных инженеров на соответствие этим стандартам, обеспечивать поддержку корпоративных и правительственных образовательных программ в области системной инженерии.




INCOSE разработало этический кодекс системных инженеров (https://www.incose.org/about/ethics.aspx).

В год INCOSE проводит два больших международных мероприятия, на которых очно встречаются системные инженеры со всего мира: международный симпозиум по системной инженерии и международный семинар по системной инженерии. Но главное, конечно, это региональные профессиональные встречи, которых проводится огромное количество (https://www.incose.org/newsevents/events/index.aspx).

Деятельность INCOSE проходит главным образом в рамках деятельности региональных (уровня штата или города в США, страны или даже группы стран в Европе и Азии) отделений.

Есть Русское отделение INCOSE (INCOSE Russian Chapter). Русское — это название по языку, а не по стране. Члены этого отделения из России, Украины, Белоруссии. Это похоже на Немецкое отделение (INCOSE German Chapter), где члены из Германии и Австрии. Русское отделение входит в INCOSE EMEA sector (отделения INCOSE из стран Европы, ближнего Востока и Африки).

Русское отделение INCOSE проводит заседания раз в две недели, на этих заседаниях заслушиваются доклады о современных практиках системной инженерии, прорабатывается использование русской терминологии, ведётся обсуждение исследований. Видеозаписи этих заседаний и демонстрирующиеся там слайды можно найти на http://incose-ru.livejournal.com/. Раз в год в подмосковном Бекасово проходит трёхдневная Рабочая встреча Русского отделения INCOSE по проблемам системной инженерии.

## Можно ли научить творчеству?

Знания системной инженерии в голову укладываются ступенечками, от простого к сложному. Как выделить такие ступеньки? Что нужно тренировать в мозгу, какую непривычную мозгу мыслительную работу делать привычной?

Тут я должен ввести понятие контринтуитивности. Мы живём в интуитивно понимаемом мире. Наши мозги ездят по интуитивным, невесть откуда взявшимся мыслительным рельсам в очень известном направлении, как трамвай – одним и тем же маршрутом. Мы родились, постепенно откуда-то у нас эти рельсы в мозгу проложились, и мышление по ним ездит, и ездит обычно мимо удобных способов решения задач, делая невозможным решение задач сложных.

Вот, посмотрел в окно – а там земля плоская. Когда мне говорят, что Земля круглая, я что отвечаю? Я говорю: «это неправда, посмотрите в окно». Мне отвечают: «вы что, Земля круглая, потому что если посмотреть за горизонт…, а я упорствую: «Вы мне рассказываете много всего лишнего, что там за горизонтом, за горизонтом я и сам вижу, что ничего нет. Вы мне про горизонт и что за ним, а я про Землю, вот же она – Земля плоская». Смотрите: вся моя жизненная интуиция показывает, что Земля плоская, я по ней хожу, вот этими ногами хожу и уж ноги-то точно знают, что Земля не круглая! Но каким-то людям, которых заботят масштабы не только 10 километров, но и 1000 километров, в голову откуда-то приходит мысль про «Земля – круглая», они начинают так мыслить. Через некоторое время выясняется, что кроме Земли ещё и Космос с его вакуумом есть, космические корабли там летают «всё время падая, но никогда не падая». Вот это я уже понять не могу, потому что при идее плоской Земли летание космических кораблей по кругу с достаточной скоростью, чтобы не падать никогда – это понять невозможно. Мысль о круглой земле контринтуитивна, она соответствует «народной теории» (folk theory), она нетехнологична.

Слово «контринтуитивность», в котором можно услышать «антинародность», важно. Каждый раз, когда появляются проблемы с пониманием того, как работают гении, обладающие каким-то искусством, которое никто не может понять и повторить («технэ» — это ведь повторяемость результата прежде всего), можно ожидать найти что-то глубоко контринтуитивное. Трамвай мысли у гениев идёт по совсем другим рельсам, нежели проложены в мозгу большинства людей. Проложить эти новые рельсы и пустить по ним свой мозговой трамвай обычно очень трудно.

Гений почему-то, сам часто не осознавая, сделал что-то совсем не так, как все остальные, он просто начал что-то делать в противоречии с интуицией всех остальных, и у него начало получаться. А все остальные действуют интуитивно, «по-народному», «как все», и у них не получается. И пока на уровень сознания гения, или тех людей, которые пытаются отмоделировать мышление гения, не вышло, в чём именно эта контринтуитивность, вы не можете передать это знание другим людям, не можете никого этому научить.

Вы не можете научить системного инженера, если вы не знаете на уровне сознания, что он делает. Вы не можете человека научить стать просветлённым за определённое время, если вы не понимаете, в чём именно содержание просветления. Чем отличается искусство от технологии? В искусстве — один раз свезло, вдохновение было, получился шедевр. Другой раз не свезло, вдохновения нет, не будет шедевра. В инженерии мы так не можем, нам нужно работать, нам нужны технологии, дающие неизменно превосходный результат.

Меня учили в детстве: никогда не работай с художниками, всегда работай с архитекторами. Почему, знаете? Художник – он всегда ждёт вдохновения, у него вот такой подход к своему творчеству, и его именно так учат. А архитектор получает абсолютно то же самое образование в смысле всех творческих техник, но его приучают, чтобы он сдавал работы в срок, чтобы он учитывал техзадание, что нужно приходить на работу, а рабочий день начинается во столько и заканчивается

во столько. С архитектором ты будешь уверен в результате, который будет соответствовать каким-то критериям качества, ожиданиям заказчика, и будет получен вовремя. А рисуют архитекторы обычно не хуже художника, так что и с этим всё в порядке.

Я дисциплине ума, наличию рельс для мышления уделяю много внимания – как в системной инженерии, так и любой другой дисциплине. У художника вроде как дисциплины ума нет, и его искусство передаётся как? Ученик смотрит на десятки, тысячи, сотни работ, учится ловить сленг профессионалов, смотрит, как работают настоящие мастера с ними рядом. Далее у трёх из десяти учеников в голове появляются какие-то правильные рельсы для трамваев их профессиональных мыслей. А у семи из десятка – не появляются. Ибо это не обучение в классическом смысле слова.

Мне-то надо, чтобы девять из десяти могли научиться (я вполне могу представить, что будет один неспособный идиот на десяток человек). Это означает, что я должен взять для обучения такое контринтуитивное знание, которое само не может быстро прийти в голову ученикам, сделать его минимальное компактное описание, а затем его каким-то образом передать ученикам. Вопрос: бывает ли такое в тех областях, которые традиционно считались «искусством», и которым считалось, что нельзя научить рационально? Да, бывает, сплошь и рядом!

Когда вы находите правильные объекты и правильные мыслительные операции, и правильные упражнения – то ученики после обучения даже не будут понимать, что им было трудно делать до обучения. Они будут неспособны вспомнить, по каким рельсам катилось их мышление до обучения, и поэтому они будут изумляться поведению необученных новичков, включая собственное поведение в период до освоения той или иной практики. Спросите моего ребёнка, почему он очень плохо умножал всего год назад – он не сможет объяснить, почему. Сейчас умножение для него вполне естественно, и не требует напряжения всех его умственных сил, как это было год назад.

## Метанойя — не просто обучение, а смена способа мышления

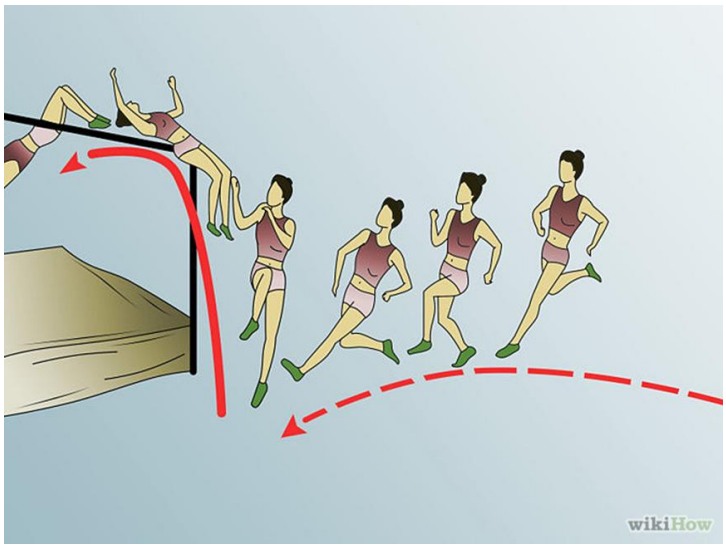
Назовём это свойство прохождения какого-то порога понимания метанойей. Слово удивительное, попробуйте его написать в разных падежах, да ещё и во множественном числе, получите очень интересные эффекты. Это слово пришло из религиозных практик и означает «перемену мыслей», полный разрыв прошлого и текущего мышления. Ты занимаешься, занимаешься в какой-нибудь семинарии, и вроде как мышление у тебя не так поставлено, как это ожидают от тебя священники. Потом вдруг в какой-то момент щёлк – и ты демонстрируешь всем, что вот у тебя такое же мышление, как это принято у священнослужителя, с этого момента ты «настоящий», а не притворяешься. Вот слово это – метанойя, такой малый западный вариант просветления. Слово "метанойя" рекомендовал использовать и гуру менеджмента Питер Сенж вместо слова "обучение", ибо слово "обучение" с его точки зрения уже совсем затасканное и не означает коренную смену образа мышления в результате обучения.

Когда метанойя произошла, то в новом состоянии мозгов человеку совершенно непонятно, в чём была проблема в старом состоянии мозгов. Представим: я знаю, что земля плоская, я долго спорю, что земля никак не может быть не круглая, но меня в какой-то момент убеждают. И я каждый раз в своих действиях сначала действую, как будто земля плоская, потом усилием воли вспоминаю, что рационально вроде бы она должна быть круглая, потом делаю это уже на уровне

рефлекса, и вижу тысячу свидетельств этой круглости Земли. И вот в этот-то момент я не могу понять, почему я считал, что земля плоская. Рационально-то вспомнить, что я так считал, я могу. Но понять, как я именно перешёл из состояния знания «интуитивной теории» в состояние владения «контринтуитивной теорией» я не могу. И поэтому не могу осознать те учебные действия, которые нужны для того, чтобы я добивался этой метанойи круглости Земли у своих учеников. Работа по составлению правильных упражнений для такой метанойи – это трудная работа, и приведённые мной ниже примеры школьной алгоритмики, просветления, тайцзицюаня показывают, что создание адекватного учебного курса вполне может занять пару-тройку десятков лет, а то и тысяч лет. Это в полной мере относится и к системной инженерии.

Особое внимание нужно обратить на то, что речь идёт об обучении не любым практикам, но «контринтуитивным», которым мозг сопротивляется особо, он же в этом случае «интуитивно знает», как должно быть, и активно сопротивляется новому знанию! Заново чему-то обучить много легче, но если уж вам свезло подхватить «народную интуицию», то научить вас чему-то более эффективному новому будет весьма проблемно: вам придётся пройти метанойю, а это требует специально организованного моделирования предметной области, последовательности упражнений и времени для прохождения этих упражнений, а также недюжинной воли – ибо вся интуиция учеников будет показывать, что учат-то какому-то безумию! Шансов пройти эту метанойю «самоучкой» практически нет, если вы не гений.

Вот, в школе учили прыгать через планку «ножницами» — подбегаешь, и прыгаешь. Но если нужно прыгнуть очень высоко, то после разбега к планочке нужно поворачиваться спиной, и прыгать назад-вверх (Fosbury Flop, изобретение 1968 года).




Это абсолютно неинтуитивно, но даёт возможность перелетать и через двухметровую планку. Нужно огромное доверие к тренеру, чтобы вы начали тренировать такой прыжок – ибо в этот момент кажется, что много-много тренировок дадут возможность преодолевать дополнительные десятки сантиметров «ножницами» или «перекатом», что совсем не так. А потом будет метанойя: вы будете не понимать, почему вообще через планку люди ещё где-то прыгают не техникой Дика Фосбери — даже если вы уже не помните, что начало таким прыжкам положил именно Дик Фосбери, и прыгать так люди начали всего на год раньше, чем высадились на Луну в 1969 году.

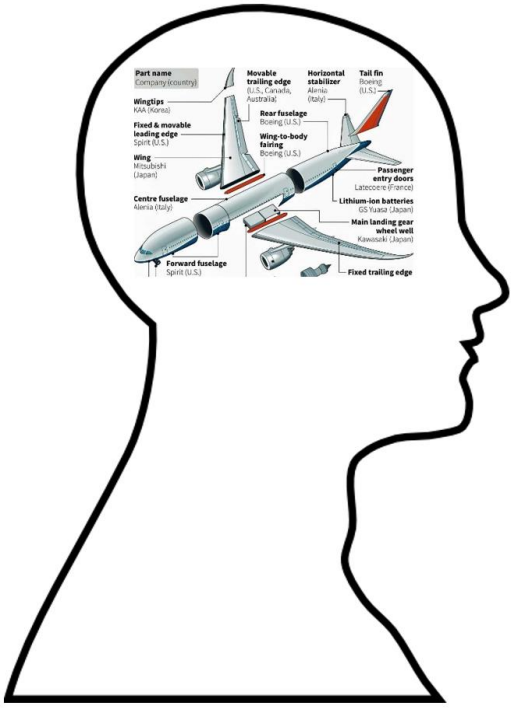
Главная метанойя системной инженерии в том, что вы начинаете думать о мире, как состоящем из систем, и описываемом многодисциплинарно для каждой системы. Если понимать систему не как “любой объект, который мы рассматриваем”, а с точки зрения системного подхода, то это крайне контринтуитивно, поэтому требует специального обучения и последующей длительной тренировки такого системного мышления.

В математике термин “интуитивный” часто подменяется термином “тривиальное” — возможность повторения "любым" в данном сообществе, а нетривиальность — невозможность повторения (спасибо за обсуждение этого вопроса математику Роману Михайлову). Демонстрация интересного нетривиального делает его тривиальным через пару тактов тренировки заинтересовавшихся, ибо в определение “интуитивности/тривиальности” и “контринтуитивности/нетривиальности” неявно входит момент времени “прямо сейчас”. Любое "контринтуитивное/нетривиальное" одного поколения становится "интуитивным/тривиальным" для другого поколения думателей.

Эту "тривиальность" вполне можно добавить в список синонимов к "интуитивности" (ибо использование "интуитивности" иногда идёт вразрез с бытовым пониманием и означает ровно наоборот — “нетривиальность, полученную инсайтом, вдохновением, озарением”).

## Можно ли научить системного инженера, или им нужно родиться?

Можно ли выучить генерального конструктора, или для этого обязательно нужно генеральным конструктором родиться, иметь врождённый талант? Ведь генеральные конструкторы прорастают из серой массы инженеров сами по себе, как трава через трещины в асфальте. Они из слесарей потихонечку-потихонечку становятся инженерами, затем зам.начальниками цехов, потом растут-растут-растут, и глянь – вот появился генеральный конструктор, как бы "сам собой". Вообще говоря, сто процентов «генеральных конструкторов», они так сами по себе и вырастали на производстве, потому что из вузов выпускали-то (да и сейчас выпускают) «просто инженеров», нигде в России не выпускают именно генеральных конструкторов, эту инженерную элиту. Просто выясняется, что у некоторых людей (генеральных конструкторов) в голове как-то чудом умещается весь самолёт, а у некоторых людей ("просто инженеров") в голову самолёт не умещается, умещается только небольшое крыло, а может даже меньше — например, только аэродинамические аспекты этого крыла, или только прочностные, вибрационные и другие механические аспекты.




Развилка в том, хотим ли мы получать системных инженеров методом охоты и собирательства (время от времени “хантить” талантливых людей, которые как-то демонстрируют особый ум и сообразительность, смекалку Кулибинского масштаба), или же мы перейдём к осёдлому земледелию в подготовке таких “генералистов” (специалистов по междисциплинарности и охвату целого)?

Честный ответ на вопрос о "выучивании на генерального конструктора": да, этому можно и нужно учить. Выученный специально генеральный конструктор будет выходить из вуза на работу и в первый же день вести себя примерно так же, как себя ведёт какой-нибудь, не очень крупный, не совсем выдающийся и талантливый генеральный конструктор, но все-таки, как ведёт себя генеральный конструктор, а не просто инженер. У него в голове будет умещаться весь самолёт как целое, а не только его небольшие части. При этом я пока опускаю менеджерские навыки генеральных конструкторов, но про менеджеров тут точно такой же разговор (можно ли научить менеджера, или это врождённый талант).

Обратите внимание: я говорю, что быть генеральным конструктором — это не врождённые способности, не талант, этому можно научить. Оппоненты говорят, что «вы же говорите немыслимое. Вот был такой человек Ойстрах, он великолепно играл на скрипке — ну, и как выучить так играть на скрипке? Ведь любая музыкальная школа учит играть на скрипке, после неё вы всегда сможете спиликать что-нибудь. Но вот выйдите после музыкальной школы на международный конкурс и спиликайте что-нибудь так, чтобы 2000 человек сказали: «Ах!»». Оппоненты, которых будет много больше, чем половина человечества, говорят, что такого «обучения на Ойстраха» не может быть, потому что не может быть никогда: «не покушайтесь на святое, это ведь человек, это его талант, это что-то божественное, это космос, который проявляется через человека, искусство трансцендентно!».

Все то же самое, что относится к обучению «на Ойстраха», относится и к обучению «на генерального конструктора». Вторая половина человечества говорит: «эта деятельность считается особо интеллектуальной, потому что мы пока не знаем, как она устроена. Но рано или поздно она станет неинтеллектуальной. Рано или поздно она будет восприниматься как обучение ходьбе – чуть ли не механическим навыкам,

несмотря на творческость и уникальность каждого отдельного шага». Первая половина человечества понятно, что на это отвечает: «уже прошло много тысяч лет сознательной деятельности людей, а решения проблемы обучения творчеству в общем виде так и не предложено – ну, и как собираетесь учить? И вообще, в СССР как-то генеральные конструкторы-самоучки тоже до космоса долетели». На что им вежливо отвечают: «до космоса-то долетели, но до Луны особо не добрались, а с марсоходами вообще у отечественных инженеров не получилось». Оппоненты отвечают «так финансирования нет». Сторонники выучивания «на генерального конструктора» говорят: «Вот именно: из-за этого очевидного неумения и финансирования нет. Кто умеет, тому и платят». И дальше напоминают, что у советских инженеров не всё хорошо было. Во времена перестройки часто шутили, что везде в мире манипулятор типа «мышь», а у нас в СССР манипулятор типа «крыс», ибо маленькие-то вещи сделать в СССР не умеют.

Когда-то я задавал вопросы, ставя в совершенно неудобное положение советских начальников: «в учебнике научного коммунизма написано, что капитализм тормозит научно-технический прогресс, особенно во время кризисов. А когда посмотришь в реальности, то обнаруживаешь, что вся лучшая аппаратура, которая нас окружает (а я работал в этот момент на ректорской кафедре, и там было много отечественной аппаратуры, и много аппаратуры из социалистического лагеря, и совсем чуть-чуть аппаратуры из капиталистического лагеря) – из капиталистического лагеря, и она – это всем очевидно! — абсолютно недостижима в плане производства её в СССР. Это ж какая скорость научно-технического прогресса должна быть у капитализма, если они во время торможения в этом самом их кризисе выпускают вот такие инженерные шедевры?».

И действительно, что отличает западных инженеров от советских, если отвлечься от обсуждения политического строя? Сторонники инженерного самоучества могут говорить, что на Западе талантливых людей держат за шкирку и выжимают из них все соки, и именно поэтому они творят лучшую в мире аппаратуру. Но я считаю по-другому: на Западе все-таки чуть-чуть научились ещё и вот это вот инженерное искусство делать не «искусством» в смысле художественного искусства, которое со вдохновением и полётом фантазии и иногда получающимся результатом, а технологией – то есть с воспроизводимым, неизменно превосходным результатом. Порядок (технология) бьёт класс (искусство).

Берём любого посредственного студента, ставим его на правильный учебный конвейер, и через некоторое время у него в голове помещается всё производство правильной «мышки». В этой «мышке», кстати, сейчас довольно много технологий: там есть и лазер, и небольшой компьютер, и беспроводное радио, и тефлоновое покрытие, которое плохо истирается, но хорошо скользит. Чтобы удержать в голове столько технологий «мышки», системный инженер должен иметь хорошо организованную голову. И к курсу третьему у него будет получаться удерживать в голове вот такую «мышку», а к пятому курсу – небольшую ракету, которая летит ещё, может, не с континента на континент, но километров 200 уж точно. Противники образования генеральных конструкторов/системных инженеров на что отвечают уже без аргументов: «вы ничего не понимаете!».

Когда-то операция деления считалась привилегией особо талантливых, и за знание деления давали докторскую степень. И итальянские университеты считались особо искусными в делении, потому что, делить в римских числах (в которых, заметим, даже нуля нет!) – это то ещё удовольствие, это трудно. Когда мне говорят, что чему-то «творческому» научить нельзя, что системный инженер, генеральный

конструктор, менеджер-организатор растут как сорная трава в столкновении с асфальтом жизни, и преподносят это как результат «производственного опыта, и только», то я отвечаю: «конечно, производственный опыт важен, но не с этого опыта нужно начинать».

### Моделирование творчества в виде, понятном даже компьютеру

Пожалуй, самым важным моделированием можно считать не просто моделирование какой-то малопонятной человеческой творческой деятельности, но моделирование её на таком уровне, что эту творческую деятельность мог бы повторить компьютер.

Французская лаборатория музыки фирмы Sony выполняет как раз такие проекты моделирования: она моделирует стили музыкантов — более того, получив общую модель музыкальных стилей, специалисты этой лаборатории добились того, что компьютер на лету обучается игре и импровизации в стиле конкретного музыканта. Это применимо не только к музыке, но и к стихам песен! Компьютер может сочинять стихи в стиле тех или иных поэтов песенников. Подробности этого проекта можно узнать тут: http://flow-machines.com/ (там видео, объясняющее суть дела: марковские цепи с ограничениями как представления понятия музыкального или поэтического стиля).

А уже упоминавшийся конкурс музыкантов-исполнителей? Результаты последнего конкурса компьютерных программ-исполнителей можно услышать тут: http://smac2013.renconmusic.org/results/. Ещё пару лет, и отличить играющий компьютер от играющего пианиста будет сложно (а непрофессионал может спутать игру компьютера и профессионального пианиста уже и сегодня).

Есть и общая теория компьютерного творчества (одновременно это и теория интереса: творчество тогда творчество, когда его результаты интересны!): http://people.idsia.ch/~juergen/creativity.html

Недаром говорят, что слово “интеллектуальный” относится только к тем деятельностям, которые люди ещё не знают, как делать. Как только любой телефон научили играть в шахматы на уровне гроссмейстера, игру в шахматы перестали считать интеллектуальной, и она превратилась в обычный спорт — перестала быть демонстрацией “ума” и “творческих способностей”.

В системной инженерии, конечно, бездна “интеллектуального” — того, что на сегодняшний день очень трудно отмоделировать, очень трудно описать в явном виде и чему трудно научить какими-то упражнениями.

Тем не менее, системной инженерии (в отличие от ненаучаемого “быть генеральным конструктором”) научить можно, и части этих умений можно научить даже компьютер.

Обзор подходов к автоматизации деятельности системных инженеров см. в http://ailev.livejournal.com/1122876.html — среди имеющихся на сегодня подходов можно выделить моделе-ориентированную системную инженерию, поиск-ориентированную системную инженерию, синтез модулей на базе теории контрактов, порождающее проектирование (generative design, в том числе generative architecture) и т.д. Компьютерный поиск (порождение, вывод, вычисление) требований, архитектуры, тестов/испытаний — это и есть следующее поколение системной инженерии, непосредственно следующее за переходом к моделеориентированности. Роль человека при этом будет сдвигаться от собственно системно-инженерной работы к методологической работе по системной инженерии. Ибо для того, чтобы объяснить системную инженерию компьютеру, нужно очень

хорошо представлять, в чём суть системной инженерии, почему она работает, и в чём отличие мышления гениального системного инженера от мышления системного инженера посредственного.

### Методология системной инженерии

Системные инженеры занимаются созданием каких-то целевых систем: самолётов, подводных лодок, медицинской аппаратуры, сетевых сервисов, небоскрёбов. Методологи системной инженерии занимаются тем, чтобы вычленить (отмоделировать: формально сформулировать и документировать) системноинженерное знание. Это нужно для того, чтобы:

- Представить это знание как объект инженерии — и перепроектировать его, сделать его лучше перед тем, как применить его в следующих проектах.
- Вычленить, структурировать и документировать знания по системной инженерии, чтобы спроектировать курсы для образования системных инженеров. Так, написание этой нашей книги — занятие не системной инженерией, а методологией системной инженерии (автор при этом находится в роли не системного инженера, а методолога системной инженерии).
- Формализовать знания по системной инженерии до уровня, достаточного для его машинной обработки, выполнения системноинженерного мышления компьютером (это часто называют инженерией знаний, но опять же — это занятия не системной инженерией, это занятия методологией системной инженерии)

Одним из недостатков советской инженерной школы был в том, что методологии не уделялось должного внимания. Опора была не на “метод”, а на “таланты”.

### Образование системных инженеров

В отражающей опыт системной инженерии NASA статье "Наука и искусство системной инженерии" (http://www.worldscinet.com/srf/03/0302/free-access/S1793966609000080.pdf), говорится, что системная инженерия наполовину “искусство” (которому нельзя обучить рационально, а можно только “прочувствовать” и к которому нужен особый талант), а наполовину “дисциплина” (близкая к чему-то бюрократическому). Это довольно точно отражает вчерашнюю и отчасти даже сегодняшнюю ситуацию, но эта ситуация (опять же неявно сравнивающая системных инженеров и музыкантов с их “искусством”) быстро меняется.

Очень спорный вопрос, можно ли учить “искусству”, и нужно ли вообще учить “бюрократии” в творческих дисциплинах (не удавит ли бюрократия творчество — это тоже ведь тема обсуждения! Не удавит ли бюрократическая дисциплина играть пальцами рук на пианино творчество игры ещё и ногами, вдруг у кого-то именно в этом будет достигнута высшая музыкальность!).

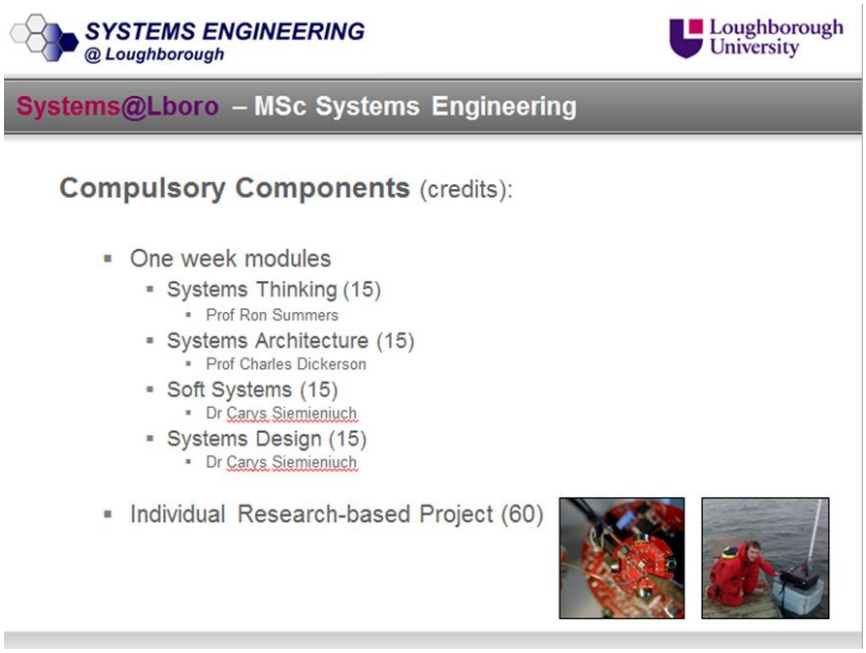
В России среди старых инженеров принято считать, что системноинженерному (т.е. охватывающему не какой-то один специальный инженерный аспект системы, а всю систему в целом) творчеству научить нельзя, “генеральные конструкторы” (проигнорируем пока тот факт, что главные и генеральные конструкторы ещё и наполовину менеджеры) по факту самоучки в этой специфической области. На Западе с одной стороны декларируют “творческость” специальности системной инженерии, с другой стороны системная инженерия — это магистерская

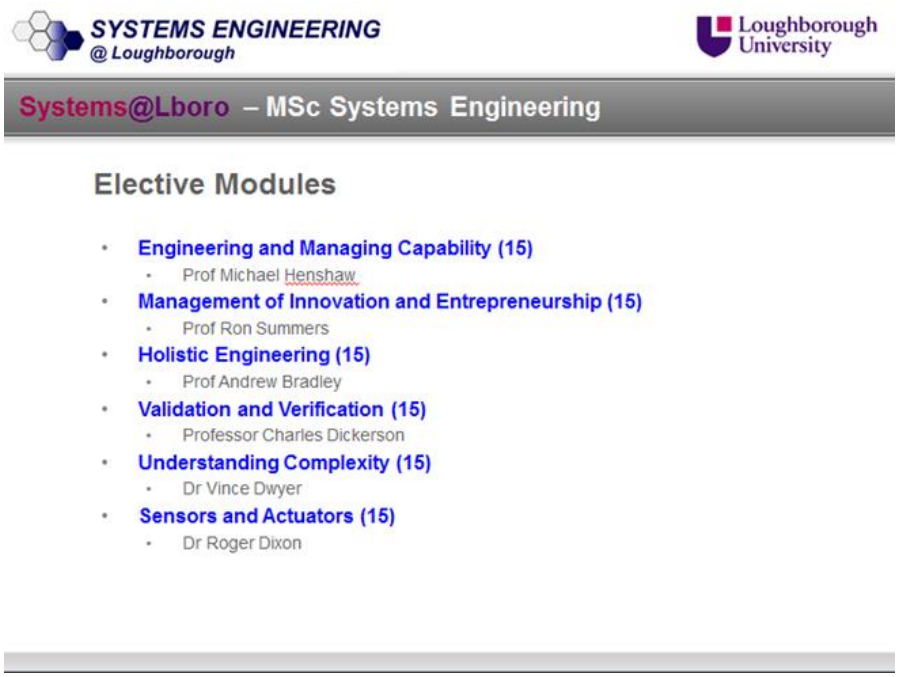
специальность в университетах.

Рекомендуется перед поступлением на курс иметь 3-5 лет опыта работы в инженерных компаниях после получения инженерной степени бакалавра. Основные требования к образованию системных инженеров изложены в прошедшем широкое обсуждение документе Graduate Reference Curriculum for Systems Engineering (http://bkcase.org/wp-content/uploads/2014/04/GRCSEv10_Final.pdf). Обратите внимание, что это именно требования к образованию: сколько, когда, кого учить, на какие работы могут рассчитывать получившие это образование люди и т.д. Требования к содержанию образования сводятся главным образом к указанию на корпус знаний системной инженерии (http://www.sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_%28SEBoK%29).

В среднем магистерский курс занимает 1800 аудиторных часов плюс примерно столько же самостоятельной подготовки, эти часы проходят за два года.

В качестве примера можно взять программу магистерского курса системной инженерии, который даёт великобританский Loughborough University, всего в этом курсе 180 кредитов (условно можно считать, что один кредит — это 10 аудиторных учебных часов плюс столько же на самостоятельную подготовку). Обратите внимание на обязательные предметы: системное мышление, системная архитектура, "мягкие" системы (системы, включающие в себя людей), системное конструирование/проектирование (в английском используется одно слово: design, создание трёхмерной конструкции материальной системы) — это всё вместе занимает 60 кредитов, и столько же времени отводится на обязательный индивидуальный учебный проект. Ну, и ещё 60 кредитов тратится на курсы по выбору (инженерные и менеджерские возможности, управление инновациями и предпринимательство, холистическая инженерия, проверка и приёмка, понимание сложности, датчики и приводы).






Учебный проект обычно выполняется командой, в которой участвуют инженеры разных специальностей. Студенты-будущие системные инженеры обычно не любят работать в такой команде: через очень короткое время выясняется, что все представители разных специальностей имеют разные интересы (например, при конструировании автономного робота нужно согласовывать прочность механики, мощность тяжёлой батарейки, мощности моторов, скорость и тем самым тяжесть компьютеров и т.д. — участники проекта каждый обосновывает необходимость его инженерных решений, несовместимых с требованиями других участников, а системный инженер вынужден решать появляющиеся проблемы). Студенту объясняется, что это и есть его работа: отныне и в будущем он всегда будет в эпицентре разработки, и его задачей как раз является решение всех проблем, возникающих от противоречий требований разных инженерных дисциплин. Его как раз учат адекватному мыслительному аппарату, позволяющему решать такие проблемы, и на этом учебном проекте он должен тренировать свои навыки системноинженерного мышления и учиться получать удовольствие от того, что он решает, казалось бы, неразрешимые проблемы.

В университетах США образование системного инженера обычно устроено немного по-другому, а в последнее время такое образование можно получить и в онлайн программах через интернет).

На магистра системной инженерии берут учить либо бакалавров физиков-математиков-химиков, но хотя бы с годом производственного опыта в инженерии, либо бакалавров-инженеров. Вся магистерская программа в США обычно объемом в 36 кредитов, но иногда бывает и 48 кредитов (http://www.coloradotech.edu/degrees/masters/systems-engineering) — и учитывайте, что кредиты обычно определяются не слишком формально. Типичный магистерский кредит обычно стоит порядка $900-$1200, а курс занимает 3 кредита и стоит порядка $2700-$3600 (типичное предложение университетской программы выглядит так: http://www.worldcampus.psu.edu/degrees-and-certificates/systems-engineering-masters/overview, цена и часы там прямо вверху справа или http://ep.jhu.edu/graduate-programs/systems-engineering, но за ценой тут нужно сходить сюда: http://ep.jhu.edu/tuition-fees). Попробуйте поиск в гугле "master of science in systems engineering" online. Вывалится огромное число ссылок на

программы подготовки магистров по системной инженерии, многие из них будут доступны и в онлайн варианте.

Кредит (credit) везде считается по-разному, в среднем это 50 лекционных минут каждую неделю на 15 недель семестра (всего 750 минут), но если это семинарское занятие, то на один кредит легко может прийтись три астрономических часа в неделю на 15 недель. При этом предполагается, что вы будете ещё тратить удвоенное количество времени на домашнюю подготовку к этим занятиям — читать литературу, решать задачки, писать эссе и выполнять мелкие проекты. Вот типичная система расчёта времени "кредитами": https://www.purdue.edu/registrar/pdf/Credit_Hour_Guidelines.pdf

Один курс — это обычно 3 кредита (хотя бывает и чуть поменьше, и чуть побольше). Это означает, что один курс займёт у вас 750минут/кредит*3кредита = 37.5 аудиторных часов + ещё два раза по столько же на подготовку, итого 112.5 астрономических часа.

Иногда пытаются упаковать обучение 15 недель в семь, это удваивает нагрузку в неделе. От работающих требуют получить разрешение от работодателя, если берётся больше двух курсов в семестр — ибо вам нужно будет потратить на занятия за 7-15 недель 225 астрономических часов, это 3-6 часов ежедневно, а "сачковать" не дадут. И семестров в году не два, а три. И учиться нужно два года. И отдать денег, как на хороший автомобиль.

Если хочется получить не сертификат, что "прослушали курс", а степень — то добавьте к этому ещё время на диплом (удвойте время на диплом, но после него вы пойдёте учиться дальше на PhD по системной инженерии) или обязательный рабочий проект.

Что же там учат, какие примерно 10-12 курсов каждый примерно в 3-4 кредита нужно пройти? Это сильно зависит от учебного заведения. Вот, например, курсы университета Джона Хопкинса при специализации именно на системной инженерии (из каталога http://ep.jhu.edu//files/2014-2015-catalog.pdf#nameddest=p76, со страницы 69):

> **Systems Engineering Focus Area**
>
> Students in the Systems Engineering focus area must meet the general admissions requirements and satisfactorily complete ten one-semester courses. In addition to the five core courses, students must also complete:
>
> 645.764 Software Systems Engineering (required course)
>
> AND
>
> *One of the following four advanced courses (after taking the five core courses):*
>
> 645.742 Management of Complex Systems
>
> 645.753 Enterprise Systems Engineering
>
> 645.761 Systems Architecting
>
> 645.771 System of Systems Engineering
>
> **Master's Project or Thesis**
>
> Students must complete either the one-semester Systems Engineering

Master's Project or the two-semester Systems Engineering Master's Thesis. The thesis option is strongly recommended only for students planning to pursue doctoral studies.

645.800 Systems Engineering Master's Project

645.801/802 Systems Engineering Master's Thesis

**Electives**

With the Systems Engineering focus area, students must complete one or two relevant electives depending on whether the student has selected the master's thesis option or the master's project.

Electives may be selected from Applied Biomedical Engineering, Applied Physics, Computer Science, Electrical and Computer Engineering, Environmental Engineering and Science, Information Systems Engineering, and Technical Management.

*Systems Engineering students may not take the following as elective courses:*

595.460 Introduction to Project Management

595.464 Project Planning and Control

595.763 Software Engineering Management

*There are two additional Systems Engineering courses that may serve as electives:*

645.469 Systems Engineering of Deployed Systems

645.756 Metrics, Modeling, and Simulation for Systems Engineering

Вот альтернативный вариант из http://www.worldcampus.psu.edu/degrees-and-certificates/systems-engineering-masters/courses — там пример сразу разбивает всё на пару курсов в семестр):

1 год 1 семестр SYSEN 510 Engineering analysis I

1 год 1 семестр SYSEN 550 Creativity and Problem Solving

1 год 2 семестр SYSEN 505 Technical Project Management

1 год 2 семестр SYSEN 531 Probability Models and Simulation

1 год 3 семестр SYSEN 522 Systems Verification Validation and Testing

1 год 3 семестр SYSEN 533 Determenistic Models and Simulation

2 год 1 семестр SYSEN 530 System Optimisation

2 год 1 семестр SYSEN 536 Desision and Risk Analysis in Engineering

2 год 2 семестр SYSEN 520 Systems Engineering (добрались, но это аж 2 год 2 семестр!)

2 год 2 семестр SWENG 586 Requirements Engineering (это вариант специализации на софте)

2 год 3 семестр SWENG 587 Software architecture

2 год 3 семестр SYSENG 594A Master's paper research

Обратите внимание на разницу в программах и попробуйте в качестве упражнения определить, из какого "системноинженерного племени" (http://ailev.livejournal.com/1158235.html) тамошние профессора.

Бывают и пост-магистровские (но меньше чем PhD) сертификации — это когда вы соглашаетесь после двух магистерских лет оттрубить ещё год, добавив 6 курсов. Вот пример таких курсов для системных инженеров в университете Джо Хопкинса:

> Required courses:
>
> 645.742 Management of Complex Systems
>
> 645.753 Enterprise Systems Engineering
>
> 645.761 Systems Architecting
>
> 645.771 System of Systems Engineering
>
> Two elective courses:
>
> 645.803/804 Post-Master's-Systems Engineering ResearchProject
>
> OR
>
> Two approved 700-level courses in the Johns Hopkins Engineering for Professionals offering

Современная инженерия моделеориентирована, и можно попробовать оценить, что это добавляет к образовательной нагрузке. Возьмём в качестве примера International Post-Graduate Specialization Diploma, которую выдаёт Ecole des Mines de Nantes (http://www.emn.fr/z-info/atlanmod/index.php/The_MDE_Diploma), правда по моделеориентированной программной инженерии, но это не меняет общих оценок. После магистрата эта программа требует 320 учебных часов плюс 6 месяцев стажировки. Обратите внимание, что системная инженерия указана как одно из мест приложения моделеориентированной инженерии — несмотря на то, что это курс именно программного (software) моделирования:

- **Module 1: Prerequisites (60h)**
  - ✓ Software Development with Eclipse
  - ✓ Free and Open Source Models for software development
  - ✓ Software Modeling
- **Module 2: Fundamentals (120h)**
  - ✓ Fundamentals of Metamodeling and DSLs
  - ✓ Theory and Practice of Model Transformation
  - ✓ Advanced Model Management: repositories & collaborative development
  - ✓ Basic Model Driven Software Development
- **Module 3: Applications of MDE (120h)**
  - ✓ Information Systems
  - ✓ Embedded Systems
  - ✓ Data Engineering
  - ✓ Web Engineering
  - ✓ Graphical User Interfaces
  - ✓ Legacy Reverse Engineering
  - ✓ Process Engineering
  - ✓ System Engineering
- **Module 4: Management (60h)**
  - ✓ Management of MDE Projects
  - ✓ Alignment of Business Needs with Technical Platforms
  - ✓ Cartography of Information Systems
  - ✓ Strategies for Information System Evolution and Modernization
  - ✓ Human and Organizational Factors in Transitioning from Previous Technologies
- **Module 5: Internship (6 months)**
  - ✓ A co-op stay in a company or in a lab to work on a MDE project.

Наша книга поддерживает магистерский курс системноинженерного мышления (для

бакалавриата материал слишком трудный). В курсе две темы (системноинженерное мышление + практики моделеориентированной системной инженерии), они занимают примерно 24 астрономических часа аудиторных занятий каждая, всего 48 астрономических часов. При этом студенты в среднем тратят на домашнюю работу (чтение литературы, ответы на вопросы, подготовку эссе) не дополнительно 96 внеаудиторных часов, а много меньше, увы. Так что классных занятий получается чуть больше, домашних по факту чуть меньше (никого ведь не выгонят за невыполнение!) — но итого получается примерный аналог западного трёхкредитного университетского курса навроде introduction to systems engineering, с результативностью прямо пропорциональной затраченному студентами времени на домашнюю работу.

Такой вводный курс системной инженерии обычно даётся как начальный в традиционном образовании магистров системной инженерии. Но очень часто он вставляется как обязательный в программы technology management, engineering management, software engineering и другие программы, где требуется общее знакомство с системной инженерией, но не специализация на ней. Эта обязательность изучения вводного курса системной инженерии для других инженерных и даже менеджерских специальностей, конечно, сильно зависит от университетов и конкретных программ. Так, в той же программе technology management университета Джона Хопкинса для специализации организационного менеджмента и инновационного менеджмента курс 645.462 Introduction to Systems Engineering необязателен, а вот для специализации проектного управления и специализации управления качеством — обязателен.

На межвузовской (в 2015г. Это МФТИ, МИСиС, МИФИ) кафедре технологического предпринимательства программа похожа на подготовку западных магистров по управлению технологиями, растянутый на пару семестров трехкредитный (в пересчёте на западные мерки) вводный курс системной инженерии обязательный и поддерживается настоящей книгой.

В России пока единственное место, где взялись готовить магистров системной инженерии — это Екатеринбург, высшая инженерная школа УрФУ (http://magister.urfu.ru/master/institutions/hse/). Не знаю, что там получится в итоге (первый набор только-только начал учиться с сентября 2014 года), но презентация программы (http://www.slideshare.net/atner/prezentacija-na-sait) там вполне сравнима с западными, а первые "полтора кредита" (как обычно в России — по аудиторным часам больше, а по часам домашней работы студентов меньше) системноинженерного мышления по данной книге студенты уже получили.

Тут нужно обязательно отметить, что в России ещё магистров учат “системному инжинирингу”, но обычно под таким названием это сводится к двум вариантам:

- Инженерия программно-аппаратных компьютерных комплексов (сервера, системное и прикладное программное обеспечение, задачи системной интеграции в информационных технологиях).
- Инженерный менеджмент, а не системная инженерия.

### *Отличия системной инженерии от других дисциплин*

#### Системная инженерия против других инженерий

Системная инженерия (systems engineering) противопоставляется “специальным инженериям” и “инженерным специализациям”. Инженерная специализация чаще

всего определяется по виду целевой инженерной системы. В этом плане различают аэрокосмическую инженерию (авиационную инженерию, космическую инженерию), сельскохозяйственную инженерию, автомобильную инженерию, биоинженерию, программную инженерию, инженерию предприятия, инженерию управляющих систем, строительную инженерию, химическую инженерию, пожарную инженерию, горную инженерию, механотронику, атомную инженерию. Инженерные специализации часто можно узнать, потому как занимающихся той или иной инженерией часто называют "отраслью" (industry).

Границы отраслей и сам принцип отраслевого деления (когда-то отраслью назывались даже не предприятия, занимающиеся в чём-то похожим производством, а предприятия, подчиняющиеся одному министерству) сегодня размыты. Иногда говорят не об отраслях, а о "промышленностях" или даже "строениях" (тяжёлое машиностроение и в нём энергомашиностроение и металлургия, точное машиностроение, среднее машиностроение и в его составе автомобилестроение, тракторостроение, станкостроение и т.д.).

Сегодня эти деления стремительно устаревают и чаще уже говорят об эко-системах (business eco-system), собирающихся вокруг каких-то целевых систем или сервисов. В эко-системах можно найти много самых разных инженерий по специальности. Так, программной инженерией занимаются и в эко-системе мобильной связи, так и в эко-системе атомной электроэнергетики.

Системная инженерия отличается от инженерий по специальности тем, что в ней используются специальные средства мышления о целостности целевой системы: средства системного мышления, когда слово "система" обозначает не просто любой объект, а именно систему — и в явном виде используются практики системной инженерии. В инженериях по специальности есть свои способы удержания целостности целевой системы в сборке различных требуемых для её создания деятельностей — но эти способы основаны не столько на общих принципах, сколько на глубоком опыте разработки тысяч и тысяч более-менее однотипных систем. Этот опыт достигается специализацией инженерной работы, обучением и воспитанием инженеров-специалистов — в отличие от "генералистов" системных инженеров.

Любой более-менее сложный инженерный проект (помним, что системная инженерия нужна тем более, чем более сложен инженерный проект) требует участия десятков и даже сотен инженерных специальностей. Разделение труда в инженерии сегодня очень глубоко, но кто ответственен за сборку всех этих узкоспециальных работ в одно непротиворечивое целое в рамках инженерного проекта? Системный инженер (или чаще — команда системных инженеров разных специализаций уже внутри системной инженерии), который имеет для этой сборки аппарат системного мышления.

В последние несколько лет азы системной инженерии стремительно изучаются инженерами разных специальностей — хотя отнюдь не везде "системные инженеры" так и называются, но методы их работы и мышление всё больше и больше соответствуют представлениям классической (двадцатилетней давности) и даже современной моделеориентированной системной инженерии. В любом случае: кто собирает работы инженеров-специалистов, занимающихся различными отдельными аспектами системы (электроникой, программным кодом, механической частью и т.д.), тот и системный инженер — его мышление междисциплинарно, в отличие от мышления инженеров по специальностям.

Так что вполне правомерно говорить о системном инженере-автомобилестроителе,

системном инженере атомной промышленности — если они используют в своей работе системное мышление и практики системной инженерии и занимаются своими системами как междисциплинарным (”межспециализационным”) целым. Если нет, то это будут просто инженеры-машиностроители, инженеры-атомщики.

Что касается “специальной инженерии” (specialty engineering), то оно относится к видам инженерной работы, которая не самая массовая в проекте. Так, если речь идёт о проектировании электроустановок, то вопросы проектирования заземления в этих установках могут быть отнесены к специальной инженерии: таких работ в проекте немного, и они требуют особых знаний, что и даёт основание их считать “специальными”, а не “общими” (http://en.wikipedia.org/wiki/Specialty_engineering).

Системная инженерия против советской инженерии

Традиционный советский способ ведения сложных разработок

А) отсутствует как отдельный предмет рассмотрения, поскольку в советское время существовал только в устной традиции (на генеральных конструкторов никто не учил специально, ГОСТы и учебники также не определяют состав работ и способы мышления генерального конструктора). Генеральные конструкторы получались методом “охоты и собирательства” (они набирались опыта на производстве — в каком-то смысле они были “самоучками”, ибо учили их только как инженеров по специальности), системных же инженеров учат этой деятельности в ВУЗах, они не “самоучки” в системной инженерии. Системный инженер имеет опыт взаимодействия с инженерами по специальностям и различными менеджерами в ходе учебных проектов, а не “по ходу дела” сразу в реальных проектах.

Б) за последние двадцать лет практически не поменялся, а ведь прошла буквально революция в способах производства — появились новые компьютерные технологии (системы автоматизации проектирования, инженерных расчётов, управления жизненным циклом), новые методы производства (из последних — 3D-печать), что нашло отражение в западных учебных программах по системной инженерии, но не нашло ни малейшего отражения в практикующейся до сегодняшнего дня советской инженерии. Речь идёт не столько изменении стандартов (ГОСТов), чтобы учесть новые производственные реалии, сколько изменении “традиций” — их и трудно изменить, поскольку они нигде не записаны, а передаются “из уст в уста”, как народный эпос или “прямая передача” от учителя к ученику как в практиках восточных единоборств. Жизнь меняется, но способ мышления “советской школы” не документирован и не меняется (в том числе в силу недокументированности даже его изменения трудно отследить).

В) включал как административное, так и техническое руководство проектами со стороны генеральных конструкторов, совмещающих менеджерскую и инженерную роли — и по мере разворачивания работ акцент с технического руководства переводился во всё более и более административный пласт. Этот тренд советского времени был поддержан в постсоветское время: ГИП (главный инженер проекта) только номинально выполняет роль технического лидера, по факту же он менеджер и несёт ответственность за соблюдение сроков, выполнение бюджета, своевременность отгрузок результатов работ и проплат вознаграждений контракторам. По факту сейчас работы по удержанию инженерного целого выполняет “консилиум” начальников отделов специальной инженерии — в ходе проведения бесчисленных совещаний и согласований результатов работы подразделений. Это вполне нормально, но методы такой коллективной работы не опираются на системный подход явно, правила согласований изобретаются по ходу

дела, нет ответственных за отсутствие "белых пятен" в целостной инженерной проработке, что ведёт к плохому качеству работ.

Г) в системной инженерии принято архитектурное проектирование/конструирование (design) как метод работы, в советской инженерии хотя и говорится о нескольких уровнях проработки проекта (эскизный/технический проект, рабочий проект), по факту это отличается от осознанной архитектурной работы. В результате на большинстве предприятий испытываются трудности с выделением и документированием архитектуры и её повторным использованием, улучшением. Такие же проблемы с требованиями: технические задания включают в себя именно задания на работу, перемежаемые с требованиями к системе, но никак не отдельный корпус требований к системе, который непрерывно уточняется и меняется по ходу разработки. В советской инженерии нет особого внимания к инженерии требований и методам этой инженерии. Также в советской инженерии нет явного различения между проверками и приёмками, отсюда огромные проблемы по обеспечению качества (приёмка неминуема, а вот проверки часто отсутствуют — и приёмка затягивается на полгода-год. Или наоборот, результаты проверки предлагается считать результатами приёмки, после чего "работы выполнены полностью, но системой пользоваться нельзя").

Увы, отдельные байки и истории успеха (типа "но мы же запустили первый в мире спутник") не убеждают в правоте сторонников развития советской школы инженерии системы в целом: как её развивать-то, где её взять, курсов-то генеральных конструкторов нет! Тем более что на каждую байку и историю инженерного советского успеха есть множество контрпримеров, их легко найти, если непредвзято (без фильтра "крымнаш") поглядеть за окошко и спросить потребителей инженерной продукции, а не поставщиков — предпочтут ли они для собственного использования изделия и сооружения российской инженерной мысли, или продукты работы инженеров из развитых других стран Востока и Запада. Пока же Индия импортировала для своей космической программы опыт системной инженерии США и Европы и с первого раза добилась успеха в запуске искусственного спутника Марса. Послевоенный выход Японии в лидеры на технологических рынках и рост качества китайского производства за последние полтора десятка лет и обеспечилось тоже не импортом советского или российского технологического знания. Хотя отдельные и конкретные технические решения вполне могли быть импортированы. Но не инженерный метод (за исключением инженерной методологии ТРИЗ, которая появилась в ещё СССР, но так и не смогла получить широкого распространения в России, развившись главным образом за пределами России).

### Системная инженерия и системотехника

В СССР в середине 60-х годов появилось несколько исследовательских групп, которые пытались разобраться с системным подходом в его научном (а не инженерном) изводе. В те времена системный подход очень часто обсуждался в связи с кибернетикой (наукой об управлении в неживой и живой природе), а кибернетика считалась чуть ли не синонимом компьютерной науки (computer science) и уж точно её обобщением. Это потом уже выяснилось, что кибернетика совсем не так всеобща, как может показаться, общей науки управления в живой природе так и не получилось, а управление в неживой природе свелось к очень небольшому курсу теории автоматического регулирования в инженерных ВУЗах (что-то типа "как бы выглядел регулятор Уатта паровой машины, если бы его делали

сегодня с использованием компьютеров”).

В 1962 году в СССР была переведена и издана книга Г.Х.Гуд, Р.Э.Макол “Системотехника. Введение в проектирование больших систем). В США эта книга вышла в 1957 году как Harry H.Good, Robert E.Machol, “Systems Engineering. An Introduction to The Design of Large-Scale Systems”. При переводе на русский язык “системная инженерия” в 1962 году стала “системотехникой”. Но этого было мало. Новой “системотехникой” ровно потому, что она была приложением системного подхода, больше всего заинтересовались кибернетики, которые в те давние времена занимались всяческими АСУ (автоматизированными системами управления). Тем самым системная инженерия попала в СССР главным образом к “автоматизаторам”, компьютерщикам, и начала развиваться с сильным “компьютерным” акцентом, и акцентом именно на системы управления (АСУ и АСУ ТП).

Это происходило и с военной ветвью: системотехника активно развивалась в том числе в военной сфере, но опять же — главным образом в системах, где существенной была вычислительная компонента (например, при разработке стратегических радиолокационных станций).

С началом перестройки, с открытием железного занавеса, из-за рубежа хлынул поток информации о тамошней компьютерной технике, и отечественные наработки по системотехнике практически смыло. Тем самым системная инженерия в существенно искажённом (хотя это можно рассматривать и как в существенно передовом виде — ибо современная системная инженерия именно сейчас активно развивается в плане слияния с программной инженерией и инженерией систем управления) просуществовала в СССР примерно двадцать лет, а потом исчезла вместе с окончанием эпохи АСУ.

Более подробно историю системотехники в СССР и её связи с системной инженерией можно узнать из доклада В.Батоврина на 85 заседании Русского отделения INCOSE (http://incose-ru.livejournal.com/45877.html).

### Системная инженерия и менеджмент

Прежде всего нужно отметить, что менеджмент сам по себе неоднородная дисциплина. Менеджментов, тесно связанных с инженерией, выделяют три (но это только самое грубое деление):

- Инженерный менеджмент (engineering management), в основе которого лежит операционный менеджмент (исследования операций, логистика, проектное и процессное управление)
- Технологический менеджмент (technology management, можно перевести и как “технологичный менеджмент” и “менеджмент технологий” в зависимости от проставляемого акцента) — стратегирование (предпринимательство, инновации) и маркетинг (продажи). Сюда же часто относят работу CTO и CIO.
- Системноинженерный менеджмент (systems engineering management) или в российской традиции “управление жизненным циклом”: часть системной инженерии, обеспечивающая целостность и преемственность в выполнении всех необходимых работ по мере того, как система проходит стадии замысла, проектирования, воплощения в жизнь, использования и вывода из эксплуатации.

Кроме этого “менеджер” ещё и работает с людьми (leadership) — выполняет роль

лидера, катализируя сотрудничество.

Менеджером называют человека, специализирующегося во всех этих направлениях/дисциплинах — и это чрезвычайно путает, ибо "универсалов-во-всём" не бывает.

От системного инженера не ожидают, что он будет заниматься продажами и выработкой стратегии фирмы кроме как в части предоставления его инженерной экспертизы. И то верно, на магистерское обучение по специальности technology management берут как с инженерных бакалавриатов, так и с бакалавриатов менеджерских.

Магистров по инженерному менеджменту готовят из инженеров-бакалавров (но не менеджеров-бакалавров), прежде всего давая им курс проектного управления, а затем уже и другие менеджерские (но уже не инженерные) курсы.

От системного инженера часто ожидают, что он будет "управлять проектом" в целом, т.е. он будет не только "инженером проекта", но и "менеджером проекта". Для совсем небольших проектов это ещё приемлемо, но вот для больших проектов с огромным числом деталей — "водитель, если ты одной рукой ведёшь автомобиль, а другой обнимаешь девушку, то ты и то, и другое делаешь плохо". Либо ты глубоко вникаешь и делаешь три-четыре варианта плана-графика проекта, либо ты глубоко вникаешь и делаешь три-четыре варианта архитектуры системы. Если заниматься и одним и другим одному человеку, то тщательно проработать получится только по одному варианту каждого, что плохо.

Инженерную дисциплину системноинженерного менеджмента aka управления жизненным циклом часто путают с менеджерской дисциплиной управления проектами. Отсюда нелепые вопросы про "кто главнее — системный инженер или менеджер проекта". Правильный ответ на такой вопрос: погибнуть проект может и от плохого менеджерского решения, и от плохого инженерного. Менеджер не может принять хорошего инженерного решения, а инженер — менеджерского, ибо у них просто не хватит профессиональной компетенции для этого. Вопрос не в административной главности (кто кого может уволить или не пустить в отпуск), вопрос в успешности разработки в целом и в этом плане вопрос о "главности" просто недопустим. Игнорирование профессионального мнения ведёт к провалу, менеджер и системный инженер просто обязаны тесно сотрудничать. К слову сказать, и менеджер, и системный инженер в крупных проектах представлены несколькими людьми каждая дисциплина, так что вопрос о "главности" ещё более убедительно переносится в область человеческих отношений, а не определение "объективно-профессионального" превосходства одной профессии над другой.

Менеджер (управленец, ответственный за бюджет и сроки сдачи, логистику) внимательно рассматривает все риски проекта (project) и там, где он считает, что эти риски неоправданно высоки, он принимает решение добавить ресурсов (например, добавить людей в проект) и чуть увеличить сроки, чтобы было время для проработки мер по уменьшению этих рисков — или же принимает решение не ввязываться в проект вообще, или выйти из проекта, не дожидаясь его завершения.

Системный инженер (ответственный за успешность системы — прежде всего, чтобы ракета полетела, а подводная лодка не утонула) также внимательно рассматривает все риски проекта (design) и там, где он считает, что эти риски неоправданно высоки, меняет конструкцию системы (а в части системноинженерного менеджмента ещё и методы проектирования и производства) так, чтобы эти риски не реализовались.

Кто в проекте главный? Каждый главней в своей области. Управление проектами — это прежде всего про планирование и контроль выполнения работ проекта в части загрузки ресурсов, диаграмма Гантта тут самая главная. Управление жизненным циклом — это как содержательно выполнять работы (заниматься инженерией требований, разрабатывать архитектуру, делать ли прототипы, какие методы работы использовать), и тут важнее содержание работ, а не их распределение по времени и ресурсам.

Разделение "генерального конструктора, он же генеральный менеджер" на системного инженера и инженерного/операционного менеджера, а также управляющего технологиями — это общий тренд в разделении труда, когда в сложных деятельностях выделяются отдельные предметы, требующие более глубокой проработки, бОльшего объёма профессиональных знаний. В операционных кроме хирурга и линейной сестры сначала появился анестезиолог, а в современной операционной анестезиологов уже двое — каждый занимается своей частью оборудования, следит за своими параметрами состояния оперируемого пациента.

У системного инженера главное, что он понимает — это какие дела нужно сделать с определением и воплощением целевой системы. Технологический менеджер разбирается в том, есть ли вообще возможность выполнить работы: заплатит ли кто за эти работы и могут ли данной командой инженеров эти работы быть выполнены прибыльно. У инженерного/операционного менеджера главное, что он понимает — это что предприятие устроено как труба, по которой движется информация и материалы, от сырья к готовой системе на выходе. Операционный менеджер смотрит на содержание выполняемых работ только в том плане, что они требуют разнородных ресурсов (людей разной квалификации, разного программного обеспечения, разного оборудования), остальное он не успевает отслеживать. Системный же инженер не успевает отслеживать кроме собственно выполняемых технологических операций ещё и перемещение результатов этих операций к местам обработки, оплату этих ресурсов, оптимизацию их загрузки.

Конечно, системный инженер и операционный менеджер работают совместно в проекте. Но вот кто из них будет главным — это не определяется в рамках дисциплин системной инженерии и операционного менеджмента. Главные оба, только по разным вопросам. Так, менеджер проекта может принять решение об увольнении какого-то из инженеров, потому как будет считать, что имеющихся ресурсов достаточно для обеспечения нужной скорости работ. И это может крайне не понравиться системному инженеру. Но системный инженер может запросто остановить запуск ракеты стоимостью пару сотен миллионов долларов за одну секунду перед стартом — и понятное дело, это тоже может не понравиться операционному менеджеру. Управляющий технологиями может закрыть вообще выполнение проектов какого-то вида на основании того, что они не соответствуют стратегии предприятия и нет перспективы выгодно продавать результаты работ. Снять же с работы могут как одного, так и другого, равно как и третьего: реальным главным является не инженер и не менеджер, а собственник предприятия.

### Инженерный менеджмент

Вот типичные определения инженерного менеджмента:

- Инженерный менеджмент — это специализированная форма менеджмента, относящаяся к промышленной инженерии, которая касается применения инженерных принципов к деловой практике

(http://en.wikipedia.org/wiki/Engineering_management).

- Инженерный менеджер отличается от других менеджеров потому что он [или она] обладает как способностью применять инженерные принципы, так и навыки (skills) в организации и направлении (directing) людей и проектов. Он уникально квалифицирован для двух типов работ: управления техническими функциями (такими, как проектирование или производство) в почти любом предприятии, так и управления более широкими функциями (такими, как маркетинг или топ-менеджмент) в высокотехнологическом предприятии. (Daniel Babcock, 1978)
- Инженерный менеджмент — это искусство и наука планирования, организации, назначения ресурсов, направляющих (directing) и контролирующих деятельностей, которые имеют технологическую компоненту (ASEM).
- Engineering management — это проектирование (designing), эксплуатация (operating) и непрерывное совершенствование целенаправленных (purposeful) систем из людей, машин, денег, времени, информации и энергии путём интегрирования инженерных и менеджерских знаний, приёмов работы и навыков, для достижения желаемых целей в технологическом предприятии и с учётом соображений по окружающей среде, качеству и этики (Omurtag, 1988)
- Engineering management — это дисциплина, адресующая принятие и воплощение в жизнь решений в части стратегического и операционного лидерства в текущих и новых (emerging) технологиях и их влиянию на взаимосвязанные системы (IEEE, 1990 и Kocaoglu, 1991).

Есть ассоциация — ASEM, American Society for Engieering Management (http://asem.org). Она небольшая, там порядка 700 членов, по факту она международная: чуть больше сотни иностранных членов. Тамошние органы управления удивительны: они представлены главным образом представителями учебных заведений, а не промышленности (то есть "инженерный менеджер" это порождение учёных, которые пытаются описать жизнь — промышленность же не признаёт наличия "инженерных менеджеров").

Первый факультет инженерного менеджмента открылся в 1907 году в Стивенсовском институте, который сейчас славится выпуском как системных инженеров, так и инженерных менеджеров (http://stevens.edu/sse/academics/graduate/engineering-management/). В принципе, инженерных менеджеров (MEM, master of engineering management) по факту выпускают везде, где выпускают системных инженеров. На входе — бакалавры инженерии, на выходе — инженерные менеджеры. Учат инженерному менеджменту не только в США, но и в Европе и по всему миру. В России инженерному менеджменту тоже учат (в отличие от системной инженерии, которой не учат до сих пор), и довольно давно, хотя программы могут называться по-другому. Так "Высшая школа системного инжиниринга" МФТИ учит именно инженерному менеджменту, а не системной инженерии — посмотрите на их программу: http://sehs.mipt.ru/edu/magistracy/program/

В США в 2007г. даже основан консорциум приличных университетов, в рамках этого консорциума эти университеты согласуют учебные программы инженерного менеджмента: http://www.mempc.org. Вот картинка, иллюстрирующая подход этого консорциума (http://www.mempc.org/images/skillsets.jpg):


То есть берут бакалавра (BS/BE, Bachelor of Science/Bachelor of Engineering), и далее он становится инженерным менеджером, выбирая либо MEM (Master of Engineering Management), либо MBA (Master of Business Administration, классическую программу для “универсального менеджера”). И обратите внимание, что этого образования всё равно не хватает, чтобы выполнять функции успешного руководителя инженерного коллектива, но не хватает по-разному в этих двух профилях подготовки.

Так, та же Школа систем и предприятий Стивенсоновского института технологии берет бакалавра инженерии и даёт ему обязательных для получения MEM четыре курса (http://stevens.edu/sse/academics/graduate/engineering-management/):

- экономика инженерии и стоимостный анализ
- элементы исследования операций
- управление проектами для сложных систем
- проектирование и управление предприятиями-разработчиками

И пятый курс по выбору.

Можно по инженерному менеджменту и Ph.D. защитить.

Тем не менее, возникает вопрос: является ли инженерный менеджмент отдельной полноценной дисциплиной с какими-то особыми практиками, или это просто такое "вечномодное" слово для обозначения произвольно сбалансированной учебным заведением смеси из инженерных дисциплин и дисциплин MBA? Переформулирую: есть ли в инженерном менеджменте что-то такое, чего не преподают ни чистым инженерам (в том числе системным инженерам), ни менеджерам (которые MBA или MSM, Master of Science in Management)?!

Это можно попробовать узнать, ежели поглядеть на Engineering Management Body of Knowledge, используемый для профессиональной сертификации. На верхнем

уровне мало что видно, ибо области знаний включают невинные с любой точки зрения:

- Market Research, Assessment, and Forecasting
- Strategic Planning and Change Management
- Product, Service and Process Development
- Engineering Projects and Process Management
- Financial Resource Management
- Marketing, Sales and Communications Management
- Leadership and Organizational Management
- Professional Responsibility, Ethics and Legal Issues

Дьявол, очевидно, в деталях. При попытках поглядеть реальные программы сразу бросается в глаза огромное количество численных моделей, как в экономическом мейнстримном образовании. В чисто инженерные дисциплины (например, организационную инженерию, как часть системной инженерии) эти области знания тоже не упакуешь — туда уйдёт лишь часть. Не пройдёшь тут и на игнорировании слова "management" (слово “менеджмент” часто можно выкинуть из названия дисциплины без особого ущерба, см. подробнее http://ailev.livejournal.com/632128.html).

Тем не менее, в инженерном менеджменте есть наличие следующих специальных тем:

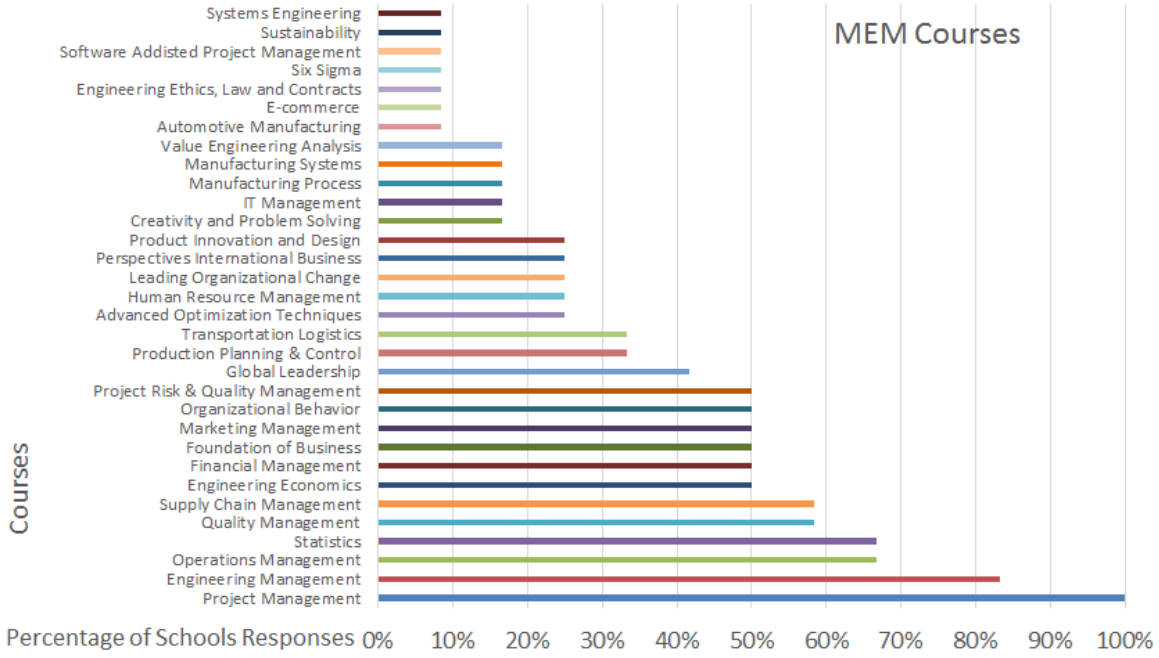
- операционный менеджмент, причём не только на уровне размахивания руками, но и на уровне factory physics (вот тут я констатирую, что в MBA этому практически не учат — http://ailev.livejournal.com/462573.html). В программах MEM хвастаются тем, что управлению проектами посвящается больше учебных часов, чем в программах MBA/MSM. Заодно отметим, что при специализации на этой дисциплине получаются operations engineer и operations egineering (это предмет отдельного разбирательства, так как это еще и связано с устоявшимся industrial engineering, традиционно понимаемым как analysis, design and control of materials, work and information in operating systems, а затем расширившимся в сторону банковских сервисов и прочих "не-промышленностей" — http://ioe.engin.umich.edu/overview/). Учтите, что operating system тут нельзя переводить “операционная система” (типа Windows или Android). Operation — это “операции”, “функционирование”, “проведение работ”, а иногда и “эксплуатация” (например operation как стадия жизненного цикла системы вполне может переводиться как “эксплуатация”).
- менеджмент технологий (смену поколений технологий — как её проводить). Это ключевая функция, поэтому иногда даже говорят engineering and technology management, вынося это на самый верхний уровень. В число основных альф инженерного проекта входят “технологии” — и это неслучайно, что они не в дисциплине инженерии! Этим менеджментом технологий обычно занимаются CTO и CIO. И очень часто это путается с технологическим менеджментом, который на грани не с инженерией, а с предпринимательством.
- design management (не только "художественного design" типа

http://www.pratt.edu/academics/art_design/art_grad/design_management/ или http://www.scad.edu/design-management/ — такие курсы после успеха Apple и опыта автомобилестроительной отрасли сейчас вставляются во все учебные программы и такой опыт требуется сейчас от всех инженеров, но и в смысле "проектирование" — http://www.plm.automation.siemens.com/en_us/products/velocity/solidedge/overview/design_management.shtml, http://en.wikipedia.org/wiki/Plant_Design_Management_System, http://am08-saopaulo.fyper.com/, хотя как отдельная дисциплина design management ещё не устоялась)

- организационные катастрофы (organizational accidents), типа расследований крупных аварий и последующего принятия мер: углублённо, как прямая связка инженерных и организационных решений.

И, в инженерный менеджмент конечно, включаются традиционные для "чисто менеджмента" темы работы с людьми (leadership), ведения бухгалтерии и управленческого финансового учёта, маркетинга и многого другого "как в MBA".

Тем не менее, в инженерном менеджменте основа – это проектное управление: планирование и выполнение проектов. Вот диаграмма о представленности разных курсов в учебной программах инженерного менеджмента разных университетов США (http://www.engineering.com/Education/EducationArticles/ArticleID/6977/What-Courses-to-Expect-in-an-Engineering-Management-Program.aspx):




## Управление технологией

Управление технологиями – это тоже учебная дисциплина, в которой иногда присутствует системная инженерия. Обычно это когда каким угодно (а не только инженерным -- входных ограничений на этот счёт нет) бакалаврам пытаются рассказать что-то про менеджмент, делая упор на инновации и предпринимательство (стартапы, но иногда и внутрикорпоративное предпринимательство) – обсуждая конкуренцию и конкурентоспособность, оценку технологий и оптимального времени перескока с технологии на технологию, финансы новых бизнесов, корпоративные слияния и поглощения с целью прихвата новых технологий и прочие подобные темы.

Почувствуйте разницу в акцентах между engineering management и technology management: в одних упор на project и операции, а в других venture и инвестиции – при всей возможной похожести необходимых для успеха дела знаний, разница существенна, ибо у проекта есть заказчик-плательщик, а у venture есть основатель-предприниматель. План проекта и бизнес-план -- в обоих говорится про деньги и время, но говорится по-разному, типовой набор стейкхолдеров и их интересы различаются.

Ещё что нужно знать, так это разницу понимания слов "innovation" в России и на Западе. В России инновацией называют любую новинку, а на Западе это будет invention (изобретение). Инновация же на Западе – это то изобретение, которое прошло стадию research (не научное исследование! Это изобретательство! Лаборатории Белла и Эдисона, не лаборатории Ферми и Кюри) и даже рабочего прототипа, и успешно вышло на рынок. Если не вышло, то это никакая не инновация, ещё просто изобретение. Не путайте яйцо, гусеницу, куколку и бабочку. Слово "бабочка" не включает в себя гусеницу, и общего слова для всех трансформаций нет. Так и тут, бабочкой инноваций гусеницу прототипа не называют. R&D management как раз про это -- как из research (прикладных! С безумными изобретателями, а не безумными учёными из мультфильмов!) перетащить что-то в development (это уже нормальная традиционная разработка, "как обычно") и далее вывести желательно даже не в продукт, а в продуктную линию.

Типичное распределение материала между "общеменеджерскими" и "технологическими и инновационными" компетенциями для мастерской степени management of technology (вариация названия, не technology management!), все курсы трёхкредитные (http://engineering.nyu.edu/academics/programs/management-technology-ms/curriculum):

Management Core
-- MG 6013 Organizational Behavior
-- MG 6073 Marketing, Credits
-- MG 6093 Accounting & Finance
-- MG 6083 Economics, Credits
-- MG 6703 Operations Management For Knowledge-Based Enterprises или MG 6303 Operations Management

Technology and Innovation Core
-- MG 6503 Management of Information Technology and Information Systems или MG 6933 Information Technologies, Systems and Management in Organizations
-- MG 7953 Global Innovation
-- MG 8203 Project Management
-- MG 8653 Managing Technological Change & Innovation
-- MG 9503 Mot Capstone Project Course или MG 9703 Project in Strategy & Innovation Mgmt или MG9973 MOT Master's Thesis

Project management (в полном соответствии с традицией engineering management) дают не в management core, а в technology core. Остальное в technology core -- вполне себе менеджерское, только с добавкой слов technology (иногда information technology) и innovation (скажем, у просто менеджеров это было бы change management, а тут будет managing technological change and innovation).

Вот обязательные курсы для technology management executive master of science (то бишь магистр-начальник управления технологиями) школы продолжающегося

образования Колумбийского университета (http://ce.columbia.edu/technology-management/courses):
-- TMGT K4116. Technology in the Business Environment.
-- TMGT K4115. Accounting and Finance for Technology.
-- TMGT K4125. Technology and the Law.
-- TMGT K4126. Strategic Advocacy for Technology Executives.
-- TMGT K4118. Behavioral Challenges in Technology Management.
-- TMGT K4120. IT and Operations Management. [это про то, что делает CIO в организации]

Как и в случае engineering management, включение инженерных курсов в программы technology management редко и возможно только там, где рядом есть сильная инженерная школа, и то с оговорками. В программе technical management (обратите внимание -- вариация названия, technical, а не technology! и таких вариаций -- тьма) университета Джона Хопкинса для специализации организационного менеджмента и инновационного менеджмента курс 645.462 Introduction to Systems Engineering необязателен, а вот для специализации проектного управления и специализации управления качеством -- обязателен (http://ep.jhu.edu//files/2014-2015-catalog.pdf). Вот, например, специализация инновационного менеджмента в тамошней программе technical management (все трёхкредитные курсы -- обсуждение кредитности на примере программы по системной инженерии было тут: http://ailev.livejournal.com/1158612.html):

Required Courses for Technical Innovation Management
-- 595.460 Introduction to Project Management
-- 595.461 Technical Group Management
-- 595.465 Communications in Technical Organizations
-- 595.466 Financial and Contract Management
-- 595.468 Fundamentals of Technical Innovation in Organizations
-- 595.762 Management of Technical Organizations
-- 595.766 Advanced Technology
-- 635.792 Management of Innovation

Вот ещё вариант того же самого -- master of technology enterpreneurship Мэрилендского университета (технологическое предпринимательство, название тоже вполне типовое). Тамошние курсы (http://mte.umd.edu/):

-- Course 1: Innovative Ideas and Concept Development
-- Course 2: Strategies for Managing Innovation
-- Course 3: Business Modeling and Customer Validation
-- Course 4: Innovative Thinking
-- Course 5: Creative Design, Prototyping, and Testing
-- Course 6: Market Development and Commercialization
-- Course 7: Legal Aspects of Entrepreneurship
-- Course 8: Financial Management and New Venture Financing
-- Course 9: Corporate Technology Entrepreneurship
-- Course 10: Fundamentals of Technology Startup Ventures

Тут можно остановиться и задать вопрос -- а где же стык между инженерами и менеджерами в этих образовательных программах по technology management? Стыков таких несколько.

Как ни странно, но часто на такие программы приходят инженеры-бакалавры, которые хотят побыстрее стать начальниками и организаторами, вместо

продолжения именно инженерного образования. Так что стык делается "внутри человека", а не осознанно авторами «образовательного салата» (назовём способ образования, при котором даются самые разные курсы, в надежде что они как-то сплавятся в голове студента. Увы, эти надежде не всегда сбываются: кусочки разных курсов часто остаются в головах студентов независимыми, связи между курсами не понимаются).

Но есть и прямое научение инженерным знаниям. Например, во всяких курсах типа Creative Design, Prototyping and Testing в Мэриленском университете, да и во многих других вынужденно рассказывают кусок системной инженерии с упором на жизненный цикл и самые-самые ранние прединженерные стадии работы -- главным образом затрагивается инженерия требований и самое общее понятие об архитектуре. Ну, в курс включается системная инженерия в явном виде (как в том же самом университете Джона Хопкинса) или из неё хотя бы курс работы с требованиями к системе, как в программе четырёх обязательных курсов master of professional studies technology management Джорджтаунского университета (http://scs.georgetown.edu/departments/15/master-of-professional-studies-in-technology-management/program#concentrations):

-- Ethics in Technology Management
-- Managing Technology
-- System Requirements and Analysis
-- Financial Analysis for Managers

Сами системные инженеры про эти практики говорят как про conceptual design (в том числе MBCD, model-based conceptual design) и подчёркивают их ориентацию на операционных (business management) и стратегических (executive) менеджеров: когда ни проекта, ни системы, ни финансирования, ни команды ещё нет, что делать непонятно, а работать со стейкхолдерами уже нужно, чтобы всё это появилось. Картинка из статьи Steven J. Saunders "Return on Investment Using Model-Based Concept Design", INCOSE INSIGHT том 17 выпуск 4, декабрь 2014:

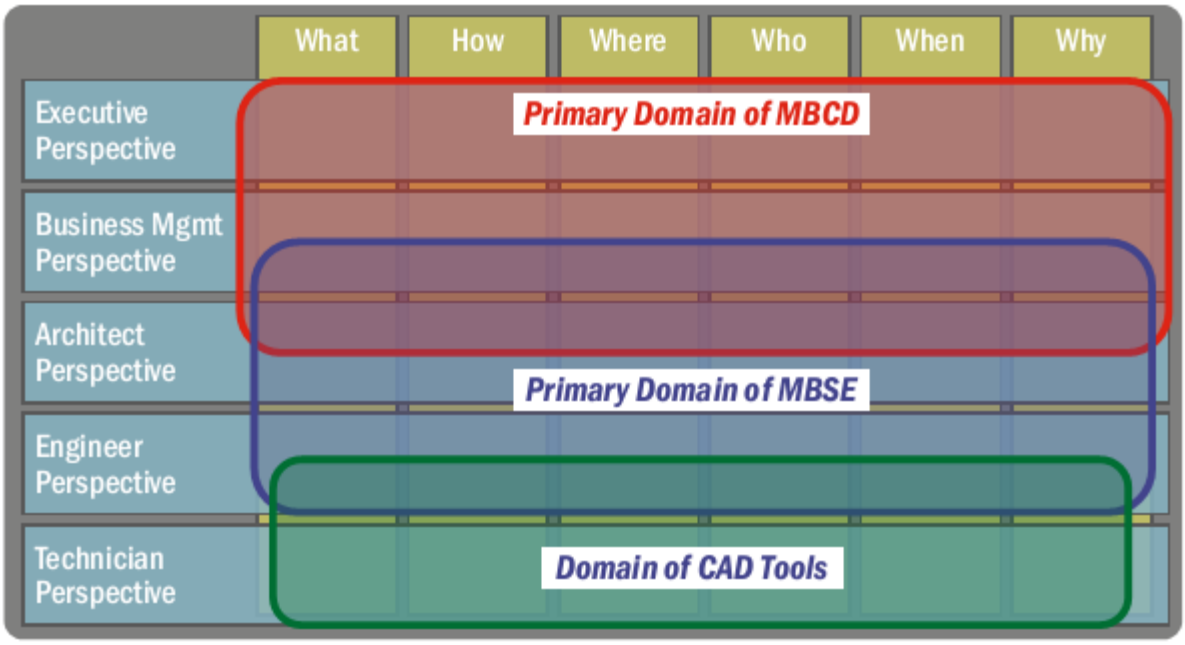



*Figure 1. Operating focus of MBCD, MBCD, and CAD tools*

## Системная инженерия и государство

Если ребёнку в руки попадает молоток, то все предметы в доме превращаются в гвозди. Если системный инженер встречается с госстроительством, то он непременно хочет им заняться. Если госстроитель (политик) знакомится с системной инженерией, то он непременно захочет её использовать. Системную инженерию в её классическом виде для целей госстроительства использовать нельзя, она предназначена прежде всего для кибер-физико-человеческих систем. В госстроительстве имеют дело главным образом с системами из людей, игнорируют кибер-составляющую (которой становится всё больше и больше в связи с распространением интернета, и скорым появлением интернета вещей), и совсем не имеют дела с физическими (”железными”) системами, с которыми системная инженерия справляется лучше всего. Краткий тут совет — если уж нужно заняться госстроительством, то используйте знания по политологии, конфликтологии, праву, экономике, социологии и системное мышление в его “мягких” вариантах, но не используйте системную инженерию, от неё будет только хуже. Государство и люди в нём — это не отсеки подводной лодки, это не детали медицинской аппаратуры, это даже не атомная электростанция вместе с её персоналом.

У проекта системной инженерии всегда есть вполне определённые стейкхолдеры, которые платят за этот проект: заказчики. Кто заказчик в госстроительстве? Политики? Чиновники? “Народ” (например, опрос общественного мнения или фокус-группа)? “Элита” (и кто её определяет)? Группы экспертов (вариант “экспертократии” — но как выбрать из этих групп “правильную”, все эксперты ведь говорят разное)? Нужно чётко понимать, что в случае госстроительства речь идёт о политике, а не о классических стейкхолдерах системноинженерного проекта. Не нужно себя обманывать, говоря, что “есть заказ, оплачивается он из бюджета, следовательно заказчиком является тот чиновник, который будет подписывать мне акт приёмки работ”. Госстроительство — это прежде всего политика, в политике подобные рассуждения неприемлемы. Более того, строить государство вы будете не из своего материала, а из всегда чужого: из других людей. Люди — это не железо, и не компьютеры. Не считайте, что именно вы из них что-то построите удачное, и не считайте, что вы как системный инженер (или программист, или сапожник) квалифицированы что-то строить из людей (таких же, как вы, между прочим). Эти люди, эти системы, из которых вы будете пытаться строить, не ваши, и не ваших заказчиков-чиновников, заказчиков-политиков.

Инженер по безопасности может защищать систему от врагов (антиклиентов). Инженер-госстроитель не может быть инженером по безопасности, разве что он строит тюрьму. Разработчики государственного регулирования все строят тюрьму, им платят именно за это, никто никогда не платит за дерегулирование. Задумайтесь над этим перед тем, как построить очередной блок государства, который дальше получит властные полномочия и употребит их для того, чтобы разрастись и получить ещё больше власти.

Нельзя также считать, что можно получить помощь от государства в развитии системной инженерии (часто об этом говорят, как о “промышленной политике”). Чиновники ничего не понимают в системной инженерии. Даже если они выделят деньги на развитие каких-либо практик системной инженерии, или создание курсов — вовсе не факт, что это будут конкурентоспособные практики, конкурентоспособные курсы. Пусть проблемы инженеров решает рынок, дело чиновников — не вмешиваться в решения рынка (не поддерживать рыночно слабых и давать им разоряться, не тормозить рыночно сильных и давать им заработать, и

не путать рыночно сильных и слабых с административно сильными и слабыми).

Нужно также очень осторожно относиться к примерам системной инженерии из военных проектов (и проектов из других областей, которые полностью зарегулированы государством — например, проектов атомной энергетики). Поскольку в этих отраслях принят принцип оплаты из бюджета “затраты плюс” (затраты, реально понесённые в ходе проекта, плюс оговорённый небольшой процент прибыли), то только полные идиоты не будут потихоньку год от года повышать стоимость проектов и сроки их выполнения, повышая тем самым и процент прибыли. Посмотрите на гражданскую технику, её стоимость, рост технических характеристик, сроки разработки за последние двадцать лет (возьмите хоть те же смартфоны: двадцать лет назад даже сотовых телефонов толком не было, не говоря уже о смартфонах) и сравните с военной техникой — сроками и стоимостями разработки. Разница будет разительна. Поэтому нужно признавать, что в военной системной инженерии есть множество интересных методологических находок, но слепо копировать этот опыт нельзя: вполне возможно, что вы откопируете заодно и прилично выглядящие способы повышения стоимости и удлинения разработки. На свободном рынке фирмы с такими методологиями бы не выжили, но на военных якобы рынках действуют совсем другие закономерности. И конечно, каждый системный инженер решает для себя сам: хочет ли он проектировать и строить машины для убийства (именно этим занимается военная системная инженерия).

Подробней на эту тему см. доклад "Системная инженерия в государственном строительстве: почему нельзя" — видео http://ailev.livejournal.com/1075745.html, слайды http://g-l-memorial.ice.ru/files/186790/ailev_syseng_may13.ppt

## 2. Формализмы системной инженерии

В этой книге мы собираемся описать дисциплину системного мышления и действия, и конкретизировать эту дисциплину для системной инженерии. Но сперва стоит обсудить – а как вообще люди описывают что-то, находящееся в мире, или же что-то, находящееся в их головах? Есть ли люди, которые профессионально занимаются описаниями? Можем ли мы позаимствовать знания из их «дисциплин описания» для области наших интересов?

Такие люди есть, и таких дисциплин можно обнаружить несколько. Описаниями мира занимаются лингвисты, логики, онтологии, специалисты по моделированию. Мы попробуем немного познакомиться с их практиками, выбирая компоненты их знаний для формирования нашего набора описательных инструментов.

### *Терминология и онтология*

Начнём мы с *терминологии* и *онтологии*.

Работа с терминологией – это работа в первую очередь с естественным человеческим языком. В каждом языке сформировались (или продолжают формироваться) наборы терминов для разных областей человеческой деятельности. И в этих областях термины приобретают значения, т.е. обозначают какие-то объекты и их отношения, см. https://ru.wikipedia.org/wiki/Терминология.

А работа с онтологией — это собственно попытка понять, как и почему мы выделяем в мире те объекты и отношения, которые обозначаются терминами. Хотя из этого

правила и есть исключения в разных научных школах, давайте запомним для нашего курса: терминология – это про язык и слова, онтология – это про реальный мир и его объекты.

### Соглашение по терминологии

Очень часто споры между людьми по самым важным вопросам жизни и смерти оказываются всего-навсего спорами о терминах. Чтобы не пропасть в таких спорах и не бояться свободы использования разных вариантов терминов для одного и того же, важно научиться различать специальные группы людей – речевые сообщества (speech communities) и сообщества значений (semantic communities). Это различение подсказывает нам стандарт Semantics of Business Vocabulary and Rules (OMG SBVR, http://www.omg.org/spec/SBVR/).

Людей в речевом сообществе объединяют естественный язык (русский, японский, немецкий и т.д.) и специальное подмножество словаря этого языка — терминология конкретной предметной области. В инженерии специальная терминология чаще всего изучается по каким-то учебникам, осваивается в непосредственном общении, или берётся из словарика определений какого-то стандарта, предпочитаемого теми или иными инженерами (ГОСТ 34.320-96, ISO/IEC/IEEE 15288 и т.д.). Поскольку разных инженеров (инженеров-строителей, инженеров-программистов, биоинженеров и т.д.) много, а, кроме инженеров, есть ещё менеджеры, юристы, кадровики, врачи и актёры — речевых сообществ даже для одного естественного языка можно обнаружить множество. У всех есть свои предпочитаемые наборы терминов из разных стандартов или учебников, и достичь однозначного соглашения по терминологии даже в области общих интересов очень трудно.

Сообщество значений (semantic commuinty) — это совокупность людей, которые одинаково понимают суть окружающих предметов и явлений. Например, все те, кто знает о существовании автомобилей и не путает автомобиль с трёхколесным велосипедом и газонокосилкой.


Когда люди общаются, они используют какую-то конкретную терминологию, выбирают слова для коммуникации. Но интересно-то обсуждать им именно предметы и явления реального мира, то есть значения терминов, их семантику. Семантика – это наука о связи разных обозначений, символов (слов из разных языков или кодов, то есть сочетаний цифр и букв) с общими для разных людей и ситуаций значениями из реального мира, поэтому мы и переводим semantic community как “сообщество значений”.

Не нужно путать “значение” со “смыслом”. Смысл текста, сообщения, иной информации определяется той ситуацией, в которой используется эта информация. Смысл – это про то, что надо делать, получив информацию, это называется "прагматика". Если семантика – про внеситуационную связь символов с их значением, то прагматика – про понимание конкретных ситуаций в деятельности. Упавшая на землю перчатка в некоторых ситуациях должна быть поднята и возвращена владельцу (владелице), но в других ситуациях такая же перчатка, упав на землю, имеет смысл вызова на дуэль.

Итак, термин — это всегда только слово. То, что этим словом обозначается в реальном мире, мы обычно называется "понятие", concept. Если люди в мире видят одинаковые понятия – они принадлежат к одному сообществу значений. А использование одинаковых терминов для определённых понятий означает принадлежность к одному речевому сообществу. Сообщество значений всегда разбито на речевые сообщества.

Никогда не видевшие автомобиль люди племени мумба-юмба вообще не входят в сообщество значений для понятия "автомобиль". Однако не знающие чужих языков люди не смогут договориться, если один будет требовать "car", а второй – переспрашивать про "машину". Но даже инженер по холодильным установкам может на секунду задуматься, когда таксист спросит его "Машина нужна?" Вспомним, что раньше (в СССР) компьютер назывался ЭВМ (электронно-вычислительная машина), а теперь уже просто компьютер. Значение не поменялось, поменялась речь — и поменялся термин, слово-обозначение.

Профессиональные сообщества часто являются и речевыми сообществами, однако терминология может существенно отличаться не только для разных профессий, но и для разных подпрофессий внутри одной профессии. То, что называется “программным средством” для системных аналитиков, работающих по ГОСТам, будет “приложением” для сейлзов иностранного софта, или "софтиной" для разработчиков.

Если невозможность договориться о терминах становится реальной проблемой, мешающей реализации проекта – к её решению есть разные подходы:

- Терминологический фашизм, когда только один термин объявляется правильным, а все остальные — неправильными (сравните с “Grammar nazi” — http://lurkmore.to/Grammar_nazi). У этого подхода есть много вариантов — безусловно требовать единственности используемого термина (отсутствия синонимов для термина), требовать ещё и соответствия принятым стандартам (определённым ГОСТам, например, а не учебникам или другим ГОСТам), требовать использования отечественного корня в слове (“мокроступы” вместо “галоши”), настаивать на соблюдении традиций (“калоши”, но никак не “галоши”).
- Терминологический пофигизм, когда на слова вообще не обращают внимания. Можно просто определять, как в математическом тексте, в начале каждого документа, “Т — ниже будет означать то-то”. Никаких “заведомо правильных вариантов” или ссылок на авторитетные источники. При этом, если значение слова меняется по ходу разговора, это часто вообще не отслеживается, речь оказывается “не строга”.
- Строгость значений с разрешением синонимии разных терминов, обозначающих одно понятие. При таком подходе обычно очень долго договариваются, какое именно понятие имеется в виду, а затем уже

используются любые слова-термины для указания на оговорённое понятие. При этом вполне допускается использование терминов, предпочитаемых разными профессиональными-речевыми сообществами. Более того, можно и не пользоваться точными терминами, если будет понятно значение. Так, при обсуждении автомобиля вполне можно обозвать его “самобеглой тележкой”, и это не будет преступлением, если адресат сообщения поймёт, о чём речь.

В нашей книге будет использоваться подход, добивающийся строгости понимания значений, при возможном использовании обозначений-синонимов. Назови хоть горшком, хоть используй пять терминов из пяти разных стандартов на трёх языках — но договорись о том, какое именно понятие/concept/значение ты имеешь в виду: собеседники должны понять не термин, а что ты под этим термином подразумеваешь.

Когда будут указываться несколько терминов-синонимов, они будут писаться через слеш: программное средство/приложение/софтина. А на то, что у каждого из этих синонимов немного разные оттенки значения, мы внимание обращать не будем.

Критика такого подхода тоже не редкость: “Как вы можете учить людей, когда одно и то же обозначаете разными словами? Вы должны выбрать один термин, и затем использовать в книге для обозначения какого-то понятия только его! Так всегда делают в учебниках!”. Ответ на эту критику прост: в жизни вы имеете все шансы встретить людей, которые обозначают понятия не теми терминами, которые введены в книгах. Так что наша книга будет вас тренировать на различение понятий и терминов: обращайте внимание – вас не просто учат новым словам, не просто заставляют зубрить терминологию. Вам стараются дать знания о понятиях и их связях. Под какими бы словами-терминами эти понятия ни скрывались.

### Выбирайте слова

Наука традиционно порождала новые термины (обозначения для появляющихся новых понятий) двумя способами:

- Бралось обычное (“бытовое”) слово, и нагружалось специальным (“научным”) значением. “Работа” в физике — отнюдь не "работа" в бытовом значении этого слова. Это самый частый способ, но он легко приводит к путанице со словами из бытовой речи.
- Чтобы сделать речь точнее, термином делалось слово, для которого в бытовой речи не было известных значений. Для этого необычное для родного уха слово бралось из иностранного языка (чаще всего — с греческим или латинским корнем) и нагружалось специальным значением. Сегодня в русском языке прихватываемым словом может быть английское слово, а не латинское или греческое — в русском-то оно бытового значения не имеет.

У нас в книге термины выбраны (в том числе при переводе иноязычных текстов — стандартов, методик, учебников) для максимизации понятности их употребления в деятельности: кто поймёт это слово, из какого он речевого сообщества? Как пользователь создаваемой терминологии отнесётся к чуждому для него жаргону “экспертов” из другого речевого сообщества?

Вот пример из проекта “Архимейт по-русски”, в котором переводилась на русский язык терминология стандарта Opengroup ArchiMate 2.0 (http://ailev.livejournal.com/988360.html). Архитектурные диаграммы для проектов информатизации бизнеса составляются айтишниками совместно с не-айтишниками (людьми из бизнеса), ибо именно не-айтишники должны решать – что в их

деятельности должно быть поддержано или изменено в ходе проекта. Окончательные решения по финансированию проектов информатизации на основании архитектурных документов принимает директор-не-айтишник. Это означает, что при переводе лучше использовать слова/термины, понятные не-айтишной части сообщества значений, а айтишники, как речевое сообщество, к этому приспособятся. Поэтому software application стало "программой" (а не "приложением"), business actor — "людьми" (а не "бизнес-агентами" или "акторами", которых по незнанию можно и с программой перепутать). Профессиональные айтишники сначала возмущаются подобным "терминологическим произволом" (ибо это термины не их речевого сообщества), но после получения опыта обсуждений с использованием "депрофессионализированной" терминологии говорят: "спасибо, такой перевод нам помог договориться".

Примерно так же мы перевели и сам термин semantic community: для специалистов из речевого сообщества лингвистов (или даже айтишников) привычнее бы звучало "семантическое сообщество", но мы попытались дать шанс что-то понять и неспециалистам из других речевых сообществ.

Вы уже обратили внимание, что тут всё время используется жаргонное слово "айтишник", а не "программист" — ибо нас заботит не только красота речи и привычные термины, но и семантика, как можно более точное указание на значения терминов в реальном мире. Ведь "программист" более узкий термин, чем "айтишник". Администратор базы данных, модельер данных, системный администратор, IT-архитектор, электронщик – все они не программисты, но айтишники. Можно было бы слово "айтишник" заменить словом "компьютерщик" — кому-то это было бы ещё понятней. С учётом всего этого мы могли бы написать программист/айтишник/компьютерщик — чтобы никому не было обидно и было бы понятней, какое значение всех этих терминов мы имеем в виду.

Бывает и так, что определённый термин, значение которого очень легко понять неправильно, уже закрепился в языке узкой профессиональной группы. Например, таков перевод "управление" для термина governance. В таких случаях в данном курсе будет использоваться наш собственный вариант, который ведёт к меньшему числу ошибок понимания. Например, governance будет переводиться как "подконтрольность", и никакие словари и стандарты тут не указ. Если какой-то процессный стандарт (например, ISO 15288) под словом process имеет в виду "практику" (в традиционном смысле стандартов ситуационной инженерии методов), то в данном курсе это будет "практика", а не "процесс". (Характерной для процессов развёртки во времени в этом "процессе" из ISO 15288 нет, там перечисляются именно "практики жизненного цикла".) Если вы попали в речевое сообщество "процессного подхода", смело используйте слово "процесс" вместо слова "практика" — но знайте, что при этом вы теряете информацию по различению процессов и практик, и речь ваша будет время от времени вызывать недоумение.

Очень часто за одним и тем же термином даже в одном речевом сообществе оказывается закреплено множество разных значений, поэтому уточнить значение даже очень распространённого термина никогда не бывает лишним. Например, Andries van Renssen выделил следующие часто используемые значения для термина "функция" (function) (читайте начиная со стр. 79 в http://repository.tudelft.nl/assets/uuid:de26132b-6f03-41b9-b882-c74b7e34a07d/its_renssen_20050914.pdf ) :

- подтип активности (поведения), процесса или события;

- некая сущность, находящаяся в определённой роли или сделанная для определённой роли;
- сама роль сущности (обычно это роль физической вещи), участвующей в активности (поведении) [Играемая роль и сущность, играющая роль — это разное! Роль - Гамлет, сущность - Высоцкий];
- указание на корреляцию, обычно как на физическую связь между какими-то аспектами: "если высота растёт, то давление падает";
- математическое отношение между числовыми объектами, определяющее их отображение друг на друга/mapping.

Ещё один пример — что может подразумеваться под часто встречающимся в информационном моделировании отношением "мета"? При обсуждении одного из языков моделирования данных (MOF, meta-object facility) было обнаружено, что слово "мета" (meta) используется в шести разных значениях, выражая шесть разных типов отношений (слайд 8 в подробной презентации http://jtc1sc32.org/doc/N1751-1800/32N1764-WG2-Tutorial-OnMOF-forSC32.ppt ):

- экземпляризация (отношение типа и экземпляра);
- группирование (отношение типа и подтипа), оно же категоризация (философская, а не из теории категорий, термин "категория" любим самыми разными речевыми сообществами, и обозначает в них разное!);
- описание (отношение описания и описываемого объекта);
- применение/стереотип (отношение шаблона и его воплощения);
- варьирование (отношение основной модели и кастомизированной);
- реализация (отношение абстрактного синтаксиса и соответствующего ему выражения).

В общем, никогда не зацикливайтесь на выбранных другими конкретных словах, слова никогда не выражают всю истину. Каждый раз пытайтесь понять, о чем в действительности идёт речь, какое значение слова имелось в виду в каждом конкретном случае. Использование терминов из стандартов не гарантирует однозначного понимания собеседником, но и использование многозначных слов не обязательно ведёт к сложностям. "Косил косой косой косой" — ведь всё понятно, не правда ли?

В этой книге не будет попыток дать точные определения и выбрать правильные термины. Мы постараемся передать понимание наиболее важных понятий и предложить разные слова, которыми их можно обозначать. На вопрос "сколько будет дважды два" будут приниматься ответы и IV, и 4, и "четыре", и four. Но не нужно обольщаться: ответы "горшок", 5, "per aspera ad astra" – приниматься не будут.

### Что такое онтология

Онтология — это и наука, отвечающая на вопрос "что есть в мире" (по-русски иногда говорят "учение о бытии", "учение о сущем"), и конкретный вариант ответа на этот вопрос. В этом она похожа на науку логику, по законам которой строятся и булева логика, и темпоральная логика, или на науку геометрию, в рамках которой развиваются теории Евклидовой или Римановой геометрий на основе разных наборов аксиом. Понимая законы онтологии, мы можем понять и 4D

экстенсиональную онтологию, и онтологию виртуальности С.Дацюка, и христианскую онтологию, хотя они предполагают мир устроенным и описываемым принципиально по-разному.

Любая онтология, определяющая, что есть в мире, должна быть как-то записана, выражена в какой-то терминологии, то есть, представлена как "онтологическое описание". В обычной речи часто путают "онтологическое описание" мира и саму онтологию. Про описание (схему нарезки мира на объекты) говорят как про онтологию (объекты, выделяемые в мире), опуская слово "описание". Разницу обычно можно понять из контекста. Мы ещё не раз вернёмся в нашей книжке к вопросам различения объекта, его определения и описания.

Философы много веков составляли очень неформальные описания мира, их книги были метафоричны, многозначны и мутны. Упоминавшийся выше Andries van Renssen заметил, что "философы прошлого недорабатывали по части строгости изложения своих философских трудов, задача получения строгого философского знания выпала на нашу долю". В 20 веке к онтологии проявили интерес разработчики программ искусственного интеллекта: их интересовало, как описывать мир настолько однозначно, чтобы даже компьютер мог интерпретировать эти описания. Они и сформулировали новое определение онтологии (вернее, онтологического описания): "онтология — это разделяемое формальное описание/представление набора понятий ("An ontology is a formal specification of a shared conceptualization", Tom Gruber — http://tomgruber.org/writing/ontology-definition-2007.htm).

Конкретная онтология (а не наука в целом!) — это один из вариантов ответа на вопрос "что есть в мире?". В общем-то, философы и логики придумали множество таких вариантов. Есть ли они вообще в мире объекты, процессы, отношения, вещи, поля? Если есть – то каковы они? Есть ли экскаваторы, торсионные поля, Гарри Поттер, философский камень, вещи, Сатана и боги греческого пантеона, биржевая котировка, благовоспитанность, справедливость, и даже философия и сама онтология? Существуют ли X=4, $E=mc^2$, гамильтониан и лагранжиан, метод конечных элементов, бит и байт, модуль упругости и его разные типы? Разные онтологии дают разные ответы на эти вопросы — а онтология как общая дисциплина изучает способы, которыми даются эти ответы.

Чем отличается семантика от онтологии?

Семантика — это про то, как мы связываем знаки/символы с означаемым, с понятиями. Символ "$" связан с денежной валютой "американский доллар". "Что означает в тексте символ $?" — это вопрос про семантику.

Онтологическая проблема — это ответ на вопрос: "что такое американские доллары?". Есть ли они в мире как отдельная сущность, явление, находится ли это явление только в наших головах – всё это онтологические вопросы.

Разумеется, в жизни очень часто путают вопросы "что означает знак X" и "что такое X". "Что такое насос?" — это спрашивают, что означает слово "насос", или спрашивают, что такое "быть насосом" в реальном мире? Пока нам достаточно научиться различать эти вопросы.

*Упражнение:* попробуйте дать ответ на онтологический вопрос — "что такое американский доллар?" [Это "риторическое упражнение": вряд ли вы дадите осмысленные ответы без знаний по теории денег и опыта онтологической работы] Можете поглядеть на список вариантов: физический предмет, абстракция, процесс,

вид товара “деньги”, валюта, фиатные деньги, единица измерения, запись на счетах. Ответьте на тот же вопрос про биткойн. Чем они отличаются онтологически?

Испытываемые вами трудности ровно того же порядка, что и у инженеров, когда им нужно определить для информационной системы в компьютере “что такое номинальный диаметр трубопровода” и как он связан с реальным диаметром, или “что такое техническое присоединение к теплосети”. А когда инженеры и менеджеры доходят до объяснения компьютеру технико-экономической модели, тогда и вопросы про американский доллар и биткойн оказываются вполне относящимся к делу.

Более подробно про онтологию и семантику при создании инженерных информационных систем можно прочесть тут: http://techinvestlab.ru/ISO15926OntologAndSemantic.

Мы пока не будем останавливаться на разнице между онтиками (наборами фактов о каком-то предмете/предметной области) и онтологиями (наборами фактов о мире в целом). Много людей называют онтики онтологиями, и пока вокруг нет маститых философов, это вполне приемлемо.

### Индивиды, классы и классификаторы

Как же людям удаётся договориться о том, как устроен мир, да ещё и формально, то есть с возможностью использовать эти договорённости для записи информации о мире и создания моделей мира в памяти компьютеров?

В первую очередь для этого надо как-то выделить и назвать то, что есть в мире, а также то, что есть в наших головах. Пожалуй, самый старый способ упорядочивать и называть вещи - это объединять их в какие-то категории, группы, разряды, классы. Самое последовательное воплощение этого принципа называется теоретико-множественным подходом, его мы и будем придерживаться в дальнейшем.

В рамках этого подхода необходимо научиться видеть сущности, категории, отношения между ними. Все сущности/объекты в первую очередь делятся на конкретные (индивиды) и абстрактные (классы и отношения).

Индивид — это индивидуальный, уникальный объект, существующий в физическом мире. Но что такое “существование в физическом мире”? Для этого есть множество философских критериев, и мы выберем самый "научный" из них. Мы будем считать, что существуют те объекты, у которых есть место в пространстве-времени (4D-подход, четырёхмерное пространство “по Эйнштейну”). Такой объект имеет некую протяжённость в пространстве (то есть размер, длину, ширину, высоту, радиус) и во времени (то есть имеет момент когда он начал существование, и момент, в который он закончил существовать). Место индивида в 4D называется "экстент" (extent), и соответствующий взгляд на мир получил название "экстенсионализм" (extensionalism). Поля и энергии мы тоже будем считать 4D объектами, физические тонкости такого подхода для нас пока не важны.

Теперь вспомните основные определения и утверждения теории множеств. Множество состоит из элементов. Например, мы можем рассмотреть множество всех индивидов x, таких, что x является автомобилем. В математической нотации теории множеств (мы сейчас начнём ей пользоваться, а потом поговорим о языках и нотациях подробнее) это можно записать так:

$$\{x \mid Car(x)\}$$

где Car(x) – логический предикат (функция. ставящая в соответствие любому объекту x значения Истина или Ложь, и истинная в том и только том случае, когда x – автомобиль). Логика и теория множеств тесно связаны, в литературе по моделированию данных, онтологии или логическим вычислениям вы встретите одни и те же математические конструкции из дискретной математики.

Имеет ли множество экстент? Мы будем считать, что не имеет, и это будет для нас определяющим признаком абстрактного объекта, не являющегося индивидом. Пусть x – индивид, но вспомните, что множество из одного индивида – это не то же самое, что этот индивид:

$$x \neq \{x\}$$

Поэтому x имеет протяжённость в пространстве-времени, но само множество – это нечто иное, это как раз абстрактный объект. Множества - это способ думать о каких-то сущностях, математические абстракции.

Вы можете встретить и иной подход, когда множество считается объединением всех своих членов, состоит из членов как из частей, и поэтому вроде бы имеет материальную протяжённость. Этот подход – иной онтологический выбор, иной взгляд на мир, и нельзя сказать, что он более или менее правильный, чем наш. Однако при рассуждениях о мире с инженерной точки зрения удобно считать, что разбиение индивидов на категории, множества – это совсем не то же самое, что разбиение индивидов на части. Разбиение на категории, выделение множеств – называется классификацией, а разбиение на части – просто инженерным разбиением (breakdown), о них вы прочтёте далее.

Итак, в рамках нашего дальнейшего рассмотрения мы будем считать, что абстрактные объекты, те, которые не являются индивидами – не имеют протяжённости в 4D, не имеют экстента.

Мы будем далее говорить не о "множествах", а о "классах", полагая пока, что это одно и то же. И это вовсе не классы из объектно-ориентированного программирования!

Отношением классификации будем называть членство объекта в классе, принадлежность элемента множеству. Стандартное обозначение того, что x – элемент множества/класса X:

$$x \in X$$

На некоторых типах диаграмм (например, язык EXPRESS-G) вы можете встретить стрелочку (направленную от класса к члену класса):

$$X \rightarrow x$$

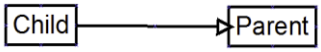
Отношением специализации двух классов — это отношение множества и подмножества (это отношение между двумя множествами, разумеется). Стандартное обозначение того, что множество/класс B – подмножество/подкласс множества/класса A:

$$B \subset A$$

На диаграммах EXPRESS-G это отображается стрелочкой с круглым концом, направленной от класса к подлассу:

A—● B

А вот в диаграммах UML то же самое отношение специализации отображается стрелочкой с треугольным концом, направленной совсем в другую сторону:




Может ли быть множество членом другого множества? Да, конечно — тогда мы будем иметь дело с множеством множеств или классом классов.

Эйфелева башня (индивид) является элементом класса БАШНЯ — это отношение классификации, членства.

Эйфелева башня ∈ БАШНЯ

Класс БАШНЯ это подкласс класса СООРУЖЕНИЕ — это отношение специализации, подмножество.

БАШНЯ ⊂ СООРУЖЕНИЕ

Но мы можем определить и более хитрый класс – класс ТИП СООРУЖЕНИЯ. Это как раз класс классов, классификатор. Классы БАШНЯ и МОСТ будут элементами класса ТИП СООРУЖЕНИЯ.

ТИП СООРУЖЕНИЯ = { БАШНЯ, МОСТ, ДОМ, … }

Один индивид может быть объявлен членом многих классов, а любой класс может быть описан и как член многих других множеств-классов, и как подкласс многих классов. В этом и состоит выразительная сила теоретико-множественного подхода. На языке классов при появлении новых знаний можно добавлять соответствующие факты к уже имеющемуся знанию, а не переосмысливать всю структуру мира, как это происходит в случае подхода с объектами и атрибутами, знакомого программистам.

Обратите особое внимание на классификаторы: классы классов. Если бы не эта конструкция, нам было бы трудно уйти от представления о том, что в мире есть главный (а то и вообще единственный!) классификатор — одна иерархия “род-вид”, что-то типа таксономического дерева Линнея в биологии. Там все живые существа классифицируются каким-то одним видом, вид принадлежит к одному классу и так далее. Такой подход, может быть, и справедлив в чистой биологии (хотя современная генетика уже ставит его под сомнение), но уж в инженерии он точно неприменим.

И в инженерии, и в иных отраслях знания разным людям нравятся разные классификаторы, разные типы родо-видовых отношений, удобные для их деятельности. Как же жить в таком мире, где нет “первичной и одной на всех иерархии классов”, как моделировать взгляды разных людей так, чтобы даже компьютеру было понятно? Очень просто: надо использовать разные классификаторы, они же классы классов. Разные взгляды на мир моделируются, прежде всего, вариантами классификаторов. А разнообразие мира внутри одного взгляда уже моделируется таксономией – деревом специализаций, отношений подмножества.

Например, классификатор продовольственных товаров не считает банан ягодой, как биологи https://ru.wikipedia.org/wiki/Банан. Классификатор Линнея — это класс классов живых существ, множество классов, выстроенных в дерево. Классификатор санитарно-эпидемиологического надзора – другой классификатор, позволяющий тоже классифицировать живые существа, но по иным признакам и в иные группы.

Ещё одним абстрактным объектом являются отношения – это пары объектов, или даже группы/кортежи из нескольких объектов. Одно отношение это отношение между индивидами, например (в виде логического предиката):

ЯвляетсяМатерью(Королева Елизавета, Принц Чарльз)

Но можно определять и классы отношений, как класс МАТЕРИНСТВО, в котором <Королева Елизавета, Принц Чарльз> - это член класса:

МАТЕРИНСТВО = {<x, y> | ЯвляетсяМатерью(x,y) }

Или класс ОТЦОВСТВО - множество всевозможных пар людей "отец-ребёнок":

ОТЦОВСТВО = {<x, y> | ЯвляетсяОтцом(x,y) }

ОТЦОВСТВО = {<Иван Иванович, Василий Иванович>, <Сергей Львович, Александр Сергеевич>,…}

Классы отношений связывают классы объектов. Так, отношение МАТЕРИНСТВО связывает класс МАТЬ (называемый областью определения отношения, domain) и класс РЕБЁНОК (называемый областью значений, range). Мы можем записать это формально с помощью кванторов и логических операторов:

$\forall$ <x, y> (ЯвляетсяМатерью(x,y) $\rightarrow$ (x $\epsilon$ МАТЬ) & (y $\epsilon$ РЕБЁНОК) )

Обратите внимание, стрелочка тут означает импликацию, а не классификацию!

В функциональной нотации это будет выглядеть так (это опять другая нотация, стрелочка теперь указывает на связь области определения и области значений функции):

ЯвляетсяМатерью: МАТЬ $\rightarrow$ РЕБЁНОК

Если отношение <Королева Елизавета, Принц Чарльз> принадлежит классу МАТЕРИНСТВО – мы можем сделать вывод о том, что Королева Елизавета классифицирована как МАТЬ, а Принц Чарльз классифицирован как РЕБЁНОК:

Королева Елизавета $\epsilon$ МАТЬ

Принц Чарльз $\epsilon$ РЕБЁНОК

Мы увидим, что при соединении теоретико-множественного подхода с подходом экстенсионализма, и ещё с 4D взглядом на пространство-время – у нас появляется возможность радикально упростить систему необходимых для описания мира отношений. Большинство интересных нам отношений оказываются подклассами очень небольшого числа принципиально разных классов отношений. Если добавить к отношениям классификации и специализации несколько видов отношении композиции (часть-целое) и отношений соединения (сторона 1 – сторона 2), то станет возможным единообразно выразить огромное число, казалось бы, совсем различных фактов о мире.

### Экстенсионализм и интенсионализм

Для различения индивидов и абстрактных объектов мы уже определили, что такое "экстент". Однако полезность этого понятия гораздо шире.

При разработке информационных систем постоянно возникает вопрос о тождественности объектов. Возникает он и для управленческих систем, и для инженерных.

Если один человек упомянул президента США, а другой – Барака Обаму, то они имели в виду одно и то же лицо? А если другие люди упомянули президента США и Джорджа Вашингтона – они имели в виду тех же лиц? В инженерии тоже нужна жёсткая логика для подобных рассуждений - описанный одним человеком насос Р-101 на схеме трубопроводов, и описанный другим человеком насос модели ПДР-15-

НШ-12 в монтажной спецификации – это один и тот же насос? А установленный в турбинном зале насос ПДР-15-НШ-12 с серийным номером RKS456/4 — как он соотносится с первыми двумя? Как описать это "в компьютере" так, чтобы и самому не запутаться, и других не запутать?

Ещё Декарт задавался вопросом: а как вообще понять, что люди говорят об одном и том же объекте, если они видят в нём самые разные свойства (то есть относят его к самым разным классам)? Скажем, один инженер говорит о высокопроизводительной системе, другой — о взрывоопасной, менеджер – о прибыльной, а финансист – о дешёвой? Как тут понять, что речь идёт об одной системе? Ответ на такие вопросы был дан в рамках философского подхода экстенсионализма ещё Декартом. В рамках экстенсионализма считают, что если экстенты, т.е. место в пространстве, у двух объектов совпадают, то это один и тот же объект. В XX веке к этому добавили ещё и протяжённость во времени, темпоральный/временнОй extent, и соответствующая теория получила название 4D экстенсионализма (4D extensionalism). Для экстенсионального подхода не важно, какие свойства и сущности увидели разные люди в объекте, или для каких применений он им нужен.

Противоположным экстенсионализму является подход субстанциализма, когда основой для определения идентичности выбирается некая неизменная внутренняя сущность предмета (substance). Ещё одной альтернативой является интенсионализм, когда для определения идентичности используется критерий наличия того или иного намерения (intent). При таком подходе микроскоп, который используется для забивания гвоздей, будет уже рассматриваться как совсем другой объект, чем микроскоп, используемый для рассматривания клетки. Однако интенсиональный подход может использоваться как дополнение к экстенсиональному, так как позволяет порождать разные точки зрения на предмет, выделять не совсем обычные предметы, определять классы.

Поначалу сугубо философская концепция экстенсионализма легла в основу ряда весьма практических подходов в компьютерных науках и в инженерии. На ней основаны:

- Стандарт ISO 15926, который считается перспективным для стандартизации обмена инженерной информацией между компьютерными системами (практически все крупные поставщики САПР заявили о его поддержке). Рекомендации по чтению для самостоятельного изучения этого стандарта см. в http://levenchuk.com/2012/10/01/iso-15926-self-education-sequence/
- Онтология IDEAS, лежащая в основе DM2 (онтологического представления для инженерных архитектурных описаний стран NATO, http://dodcio.defense.gov/TodayinCIO/DoDArchitectureFramework/dodaf20_ontology1.aspx Обратите там внимание: Individuals are Things that exist in 3D space and time, i.e., have 4D spatial-temporal extent).

Только то, что существует в мире как уникальный физический объект, имеет 4D пространственно-временную протяжённость, является онтологическим “индивидом”. Однако по данному адресу в городе неизбежно один человек увидит дом, другой увидит детский сад, третий увидит объект ЖКХ, четвёртый увидит строение, и все эти люди для целей своей деятельности будут настаивать на своём понимании и вытекающей из него классификации. Однако между ними всеми можно будет договориться на единой основе. Если обсуждаемые ими якобы разные объекты занимают одно и то же место в пространстве-времени, то это один и тот

же объект, независимо от разных имён, разного понимания его назначения, деления на части, описания свойств и т.д. Надо не забывать, что мы живём в *нашем* физическом мире, и все наши мысли и намерения в конечном итоге как-то связаны с *этим* миром, а не чем-то *потусторонним (трансцендентным)*.

В дальнейшем мы увидим, что именно такие рассуждения позволяют системным инженерам более или менее формально и однозначно работать со многими описаниями системы, и в то же самое время всегда видеть систему как целое. Такое видение системы можно назвать "многерицей" — взгляд на единую сущность, данную нам во многих очень разных на вид ипостасях. Однако все рассуждения о системе должны быть привязаны к физическому миру, пространству-времени. При осмыслении систем сначала выделяются индивиды, занимающие собой какое-то пространство-время, и только потом они осмысливаются через отношения друг с другом и с абстрактными объектами – классами разных классификаторов. Если вам начинает казаться, что разговор о системе стал слишком уж абстрактным – попытайтесь найти в физическом мире то, о чём идёт речь, и жизнь сразу наладится.

### 3D и 4D

Поговорим о времени поподробнее. Нам надо научиться различать, говорим ли мы в разные моменты времени об одном и том же объекте, или мы говорим о разных объектах, да ещё когда сами объекты как-то изменяются.

Если считать, что индивид существует только в трёхмерном пространстве, то философы не могут сказать, что происходит с ним между какими-то моментами времени. Объект существует только как "объёмные фотографии" на данный момент, трёхмерные срезы реальности. Может быть, все мы возникли лишь миг назад, и вся наша память о прошлом – фикция?

Нам, разумеется, интересны именно изменения, то, что происходит между двумя моментами времени — поэтому чисто трёхмерный подход нас и не устраивает. Представьте себе, что до какого-то момента у женщины не было детей, а потом родился ребёнок. В трёхмерном мире нам придётся сказать, что до какого момента она относилась к классу БЕЗДЕТНАЯ, а потом оказалась в классе МАТЬ. Мы будем вынуждены ввести понятие события перемещения между классами, связи индивида, класса и события, время события. Описание мира перестаёт быть единообразным и компактным, а ведь именно это было нашей целью.

Выручает переход к пространственно-временному описанию, к 4D. Но и вариантов четырёхмерного описания оказывается много.

Есть недалеко ушедшая от 3D концепция эксдурантизма (exdurantism, или Stage Theory), где в каждый момент ("слой времени") объекты мыслятся как разные, но мы соглашаемся, что эти разные объекты – как бы один объект, в котором есть эти "разные темпоральные стадии".

В конкурирующей концепции пердурантизма (perdurantism) объект считается полноценным четырёхмерным объектом, существующим во времени так же, как и в пространстве. При этом объект у нас один, но в нём выделяются другие объекты — его темпоральные части, по аналогии с привычными пространственными частями. В трёхмерном мире у моего стола есть столешница и ножки. Во времени у этого же стола есть часть от изготовления до 12:00 1 января 2015 года, есть часть от 24:00 30 декабря 2014 года до 15:00 2 февраля 2016 года, есть часть на 17:30 20 июля 2018 года. Все эти части выглядят примерно одинаково – как стол, то есть занимают

в целом одинаковую часть пространства и непрерывно в нём перемещаются, только различаются количеством пятен и царапин на них. Это и позволяет нам считать данный предмет на период его существования одним и тем же столом - в силу принципа экстенсионализма. Однако если в 19:00 23 марта 2019 года стол будет безнадёжно разрушен в мусороперерабатывающей машине – в этот момент его четырёхмерное существование (жизненный цикл) прекратится.

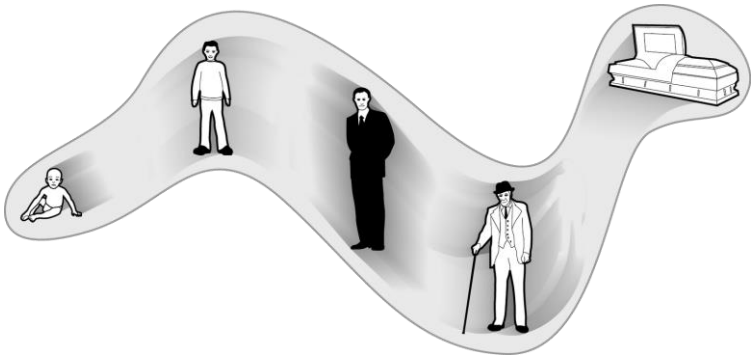
Не все философы согласны с принципами пердурантизма. Некоторым нравится эксдурантизм, им так проще отвечать на “человечий” вопрос "как вы объясните, что я чувствую протекание времени?". "Я сейчас не равен мне через пять минут" – это философский ответ на такой вопрос. Лекция философа Юрия Балашова – хорошее введение в 4D онтологии на русском языке:

> http://vic-gorbatov.livejournal.com/76485.html (первая половина лекции),
>
> http://vic-gorbatov.livejournal.com/76689.html (вторая половина лекции),
>
> http://vic-gorbatov.livejournal.com/77968.html (дискуссия и ссылки на переводы), http://vic-gorbatov.livejournal.com/78216.html (продолжение дискуссии).

Однако автор стоит именно на позиции эксдурантизма.

Специалист по моделированию данных Mathew West, профессионально занимающийся именно инженерными данными, при определении 4D онтологии для инженерных информационных систем однозначно выбирает пердурантизм: "In an information systems context, the first of these (exdurantism) is just unworkable because of the very large number of stages you might need to consider, so I adopt here the second option of parts extended in time as well as space" (p.112 в http://www.amazon.com/Developing-High-Quality-Data-Models/dp/0123751063).
Именно эта точка зрения отражена в стандартах ISO 15926 и IDEAS. В инженерии сегодня победил пердурантизм, обсуждение пространства-времени ведётся как бы “вне времени”, наблюдатель находится “извне мира”, а объекты представляются ему эдакими “темпоральными червяками”.

"Червяк" – это трёхмерное представление объекта, двигающегося в пространстве и во времени, тем самым "заметая" собой червеобразную область пространства.



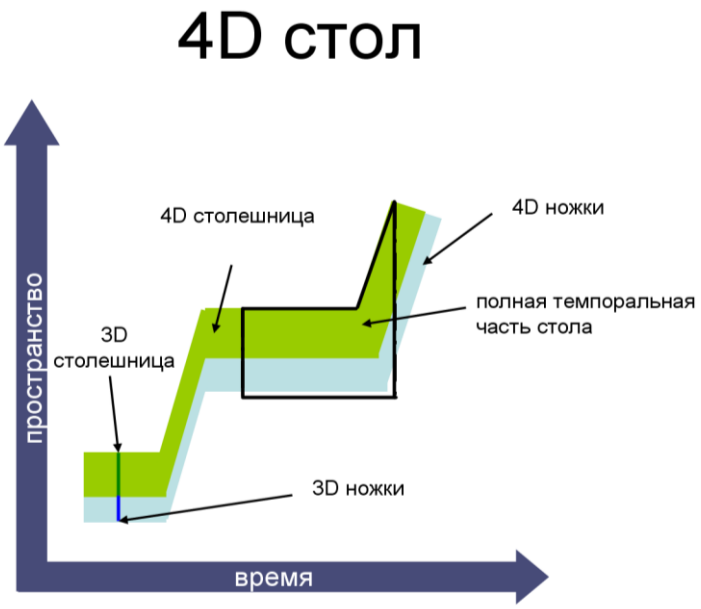
Для совсем простой иллюстрации рисуют плоскую фигуру, где все пространственные оси сжаты в одну, а время нарисовано как вторая ось, вдоль которой и движется без остановки наш объект.

Перестроить мышление с 3D времени на 4D время не так просто. В 4D есть и прошлое, и будущее, но они считаются существующими с точки зрения рассуждающего о мире "одновременно", наблюдатель стоит вне времени самого мира.

Научимся видеть на диаграмме темпорального червяка важные для нас варианты нарезания его на части. Вариантов можно выделить несколько, но все они в 4D мире описываются одним и тем же отношением - отношением часть-целое. Мы можем увидеть в объекте:

- обычные пространственные 3D части (в каждый момент времени);
- полные темпоральные части (Temporal Whole Part) – части целого, в каждый момент времени полностью совпадающие с его пространственным экстентом;
- смешанные темпорально-пространственные части.

В пространстве стол состоит из 3D частей - ножек и столешницы, ножки дополняют столешницу до полного стола, а столешница дополняет ножки. Во времени полная темпоральная целая часть стола от 24:00 30 декабря 2014 года до 15:00 2 февраля 2016 года включает и ножки и столешницу, на этом интервале никаких других частей у стола нет. Именно поэтому полную темпоральную часть так называют – чтобы подчеркнуть её полноту на протяжении определённого интервала времени. Наконец, смешанная часть - четырёхмерная столешница от 24:00 30 декабря 2014 года до 15:00 2 февраля 2016 года – тоже часть стола, но это и не обычная трёхмерная столешница, и не полная темпоральная часть. Попробуйте разобрать, как эти объекты нарисованы на следующей диаграмме.




Теперь понятно, что нам делать с примером с матерью. Именно две полные темпоральные части индивида должны быть классифицированы как БЕЗДЕТНАЯ и как МАТЬ, а индивид целиком (как говорят – на протяжении всего жизненного цикла) принадлежит только к классу ЖЕНЩИНА. Для темпоральных частей в нашем обычном языке нет названий, поэтому в формальной записи разным темпоральным частям одного индивида присваиваются условные идентификаторы, как это привычно, например, программистам. Набор интересующих нас фактов в математической нотации мог бы выглядеть примерно так (это смесь нотаций теории множеств и логических предикатов):

Марья Петровна ϵ ЖЕНЩИНА

TemporalWholePart(Марья Петровна, ind00001)

TemporalWholePart(Марья Петровна, ind00002)

Ending(ind00001, 28 февраля 2015 г.)

Beginning(ind00002, 28 февраля 2015 г.)

ind00001 ϵ БЕЗДЕТНАЯ

ind00002 ϵ МАТЬ

Одним из следствий 4D подхода является возможность рассматривать классы как "вечные", с неизменным составом. Все проблемы изменения классификации – станок включён или выключен, машина исправна или сломана, и мириады других – решаются классификацией темпоральных частей объектов. Никакой объект не переходит в рамках такого взгляда на мир из класса в класс. Просто в некоторый момент темпоральная часть подходит во времени к концу своего существования, а следующая за ней темпоральная часть уже принадлежит к иному классу.

Для полноты философской картины, лежащей в основе этого подхода к компьютерному описанию мира, нужно познакомиться ещё и с концепцией возможных миров (possible worlds, http://plato.stanford.edu/entries/possible-worlds/). Эта концепция оказывается важна при обсуждении планирования, и тем самым тесно связана с онтологическим обсуждением времени. Но тут мы этого делать не будем.

### Функциональные объекты

Теперь можно перейти к тому, как подход 4D экстенсионализма решает проблему описания объектов, изменяющихся во времени (вспомним вопрос о президентах выше).

Подход 4D позволяет работать с очень непривычными объектами, которые могут исчезать и появляться совершенно другими. Это тоже индивиды со своим экстентом, но их выделение в мире подчиняется интенсиональному принципу, то есть соответствует чьему-то представлению о функции этого индивида, намерению.

Например, я могу увидеть в своей жизни четырёхмерный индивид “моя любимая игрушка” — это плюшевый мишка в период 40 лет назад, игрушечный самолётик в период 30 лет назад и планшетный компьютер сегодня. А в промежутках, может быть, мне было не до игр, и предмет “моя любимая игрушка” вовсе не существовал. Пространственный экстент данного индивида (форма и состав в 3D) за это время несколько раз поменялись, поэтому для определения идентичности мы воспользовались его функцией, намерением пользователя.

Такие необычные объекты называют функциональными. Однако повторим, что каждый из них – обычный индивид в экстенсиональном смысле, в каждый момент времени (когда он существует) – по нему можно постучать, понюхать, уронить на ногу.

Зачем нужны функциональные объекты? Представим себе, что "президент США" – это такой четырёхмерный функциональный объект, выделенный на основе своей функции, роли в государственном управлении США. Он существует с 30 апреля 1789 года по настоящий момент, а также во многих возможных версиях будущего. При этом с 22 февраля 1732 по 14 декабря 1799 существовал обычный "четырёхмерный" человек Джордж Вашингтон. Что же происходило с 30 апреля 1789 по 4 марта 1797 года? В этот период два четырёхмерных индивида пересеклись. Полная темпоральная часть "президента США" совпадала с полной темпоральной частью "Джорджа Вашингтона". А потом они снова разошлись – следующая полная темпоральная часть "президента США" совпадала с полной темпоральной частью

Джона Адамса, потом с полной темпоральной частью Томаса Джефферсона, и т.д. Иллюстрирующая картинка взята из книги Matthew West.

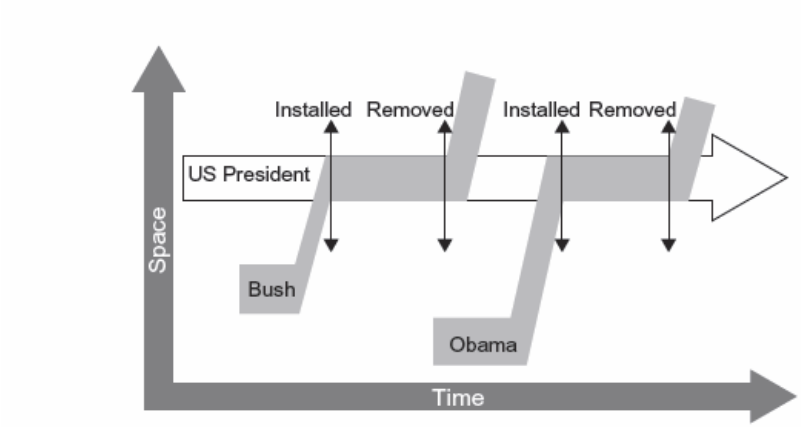
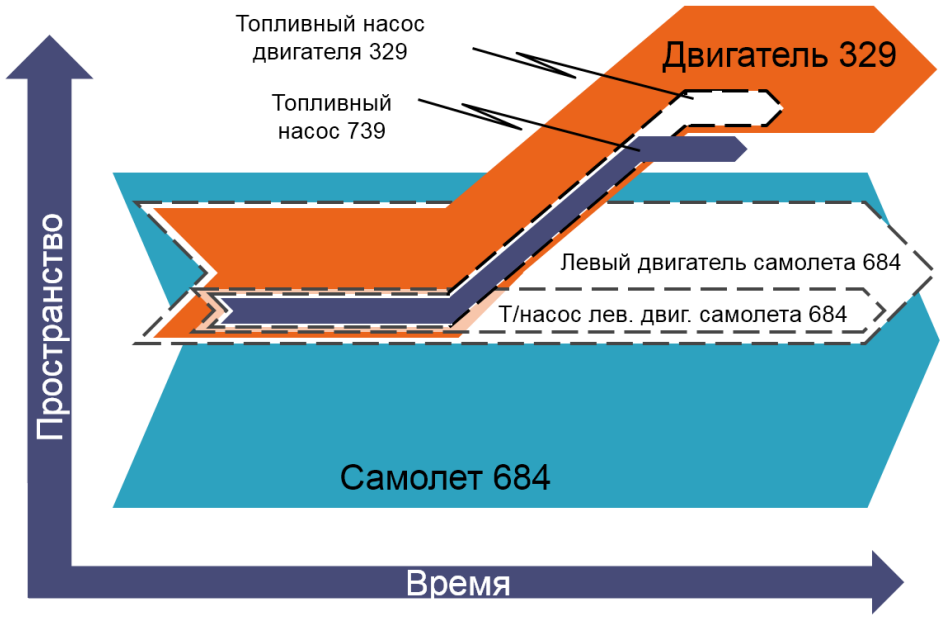



**Figure 10-2** The President of the United States.

Таково необычное поведение четырёхмерных объектов – они могут совпадать друг с другом темпоральными частями или пересекаться ими. Посмотрите на картинку (пример Matthew West) и сами разберите нарисованные на ней примеры:

## Многоуровневая пространственно-временная карта элементов системы




Двигатель 329 (обычный индивид) пересекается темпоральной частью с Самолётом 684 (обычный индивид). Двигатель 329 (обычный индивид) совпадает темпоральной частью с темпоральный частью Левого двигателя самолёта 684 (функциональный индивид). Левый двигатель самолёта 684 (функциональный индивид) является частью (то есть тоже пересекается, но во все моменты времени) с Самолётом 684 (обычный индивид).

Четырёхмерная картина мира с функциональными объектами и темпоральными частями оказывается очень удобной для описания изменений. Эти описания получаются более точными, строгими и компактными. Стандартными отношениями часть-целое мы смогли описать то, для чего при иных подходах необходимо было бы определять специальные отношения "выполнять роль", "занимать место" и т.п.

### Процессы и действия

В 4D экстенсионализме всевозможные “изменения”, “действия”, “процессы” оказываются составными четырёхмерными индивидами, состоящими из темпоральных частей всех четырёхмерных индивидов, принимающих в них участие.

Так, "танец" как индивид в какой-то момент начинает существовать, а в какой-то момент прекращает существование. Он является целым и включает в себя все участвующие в нём индивиды как части (отношение composition). Танец — это не только четырёхмерные танцоры, его исполняющие (их темпоральные части от начала до конца танца), но и поддерживающий их фрагмент четырёхмерного пола, и ещё четырёхмерный объем воздуха с колебаниями в нём, ибо в этих колебаниях — музыка для танца. Танец – это индивид особого типа, "действие". А отношение часть-целое между танцем и всеми участвующими в нём индивидами (танцорами, полом, воздухом) – относятся к особому подтипу отношения "часть-целое" – отношению "участие в деятельности".

Мы обсуждали, что по обычному или функциональному индивиду можно “постучать”. Теперь понятно, что условно “постучать” можно и по процессу, действиям, какой-то иной активности. И индивиды какого-то предприятия, и индивиды какого-то “бизнес-процесса” тем самым становятся вполне “физичными”, неабстрактными, имеющими пространственно-временную протяжённость, их легко представить. Для начала нужно просто перечислить входящие в процесс физические объекты-индивиды.

Обычно люди с трудом договариваются о “процессах” в 3D, ведь процессы, изменения очень трудно увидеть. В 4D люди договариваются об участвующих в процессе объектах, а происходящие с ними изменения описывают в терминах смены их темпоральных частей, каждая из которых представляет какое-то состояние объекта.

В 4D индивидом является каждый момент времени. Весь мир в момент времени 12:00 по Гринвичу 30 марта 2015 является одним огромным пространственным экстентом, и факт выпуска в этот момент автомобиля с конвейера может быть записан как утверждение: темпоральная граница индивидуального автомобиля является частью этого огромного индивида. Вообще любые события в жизненном цикле индивида являются границами его временных частей – вот до этого момента танца не было, а начиная с него индивид уже существует. В 4D пространстве это просто сложная поверхность, разграничивающая темпоральные части – взгляните снова на диаграммы.

Подробнее про теоретико-множественный подход, 4D экстенсионализм, многие другие связанные с этим понятия и методы работы с информацией можно прочитать в книге: Chris Partridge “Business Objects: Re-Engineering for Re-Use” (иногда её также называют BORO-book) http://www.borosolutions.co.uk/research/content/files/books/BusObj-Printed-20050531-with-watermark.pdf/at_download/file .

### О логических уровнях

Научившись выделять в мире объекты и отношения и классифицировать их, попробуем продвинуться дальше. Как выглядит процесс, в котором мы выделяем фигуры из окружающего фона?

Когда мы описываем мир, мы договариваемся об объектах в нём многоступенчато, строим его описания на нескольких последовательных уровнях. Мы будем говорить

о “логических уровнях”, нашим основным инструментом остаётся классическая логика, основанная на теории множеств. Понятие логических уровней было развито Gregory Bateson, который оттолкнулся от различения Korzybsky между картой и территорией: “идти по карте” это совсем не то, что “идти по территории”. Карта абстрагирует (моделирует) территорию. Более того, у карты есть “легенда”, которая абстрагирует карту. Надписи в легенде сделаны алфавитом, который абстрагирует надписи.

Пользуясь введёнными выше понятиями, сформулируем следующий принцип: на каждом логическом уровне описываются классы, члены которых встретятся уровнем ниже, и при этом используются классы, определённые уровнем выше.

Чтобы описать А, нам нужно использовать язык Б, чтобы описать язык Б, нам нужно отмоделировать его на языке В, и так далее (пока мы не сочтём, что какой-то из этих “высших языков” уже понятен без дальнейшего описания на другом языке). Возникают многоуровневые цепочки знаковых систем, которые находятся друг к другу в отношении “мета”. Грамматика — это мета-модель языка. Язык записи грамматик — это мета-мета-модель языка. В моделировании говорят о “мета-моделировании”, а поскольку уровней много, то о мета-мета-моделировании, мета-мета-мета-моделировании, и т.д.

Термин "моделирование" используют для обозначения отношения модели к тому объекту, который является целью моделирования (индивиду или классу), а отношения между уровнями языков моделирования называют "мета-моделированием".

Это то самое отношение "мета", которое мы обсуждали выше. Теперь, в рамках теоретико-множественного подхода, мы будем использовать только два первых значения этого термина, экземпляризацию/классификацию и группирование/специализацию. Обычно в рамках одной дисциплины рассматриваются специализации объектов, а вот сами объекты определяются “междисциплинарно”, с более высокого логического уровня, через классификацию. Иные значения "мета" в рамках нашего подхода просто сводятся к этим двум (но они могут вам встретиться как самостоятельные и описанные совсем по иному в рамках других подходов).

Для карты как модели территории легенда — это мета-модель карты. На языке инженеров или программистов это тоже легко объясняется. Если принципиальная схема холодильника – это модель холодильника, то система обозначений принципиальной схемы — это мета-модель принципиальной схемы. Определение языка SysML (Systems Modeling Language) по отношению к модели холодильника на языке SysML будет мета-моделью. Трудней подняться ещё на уровень выше: каковы “легенды”, “грамматики”, “языки задания” для этих мета-моделей? Определение языка SysML является профилем (расширением) языка UML (Unified Modeling Language), который, в свою очередь, описан на языке MOF (Meta-Object Facility, название этого мета-мета-языка говорит само за себя). Кстати, это семейство языков предусматривает до десяти уровней мета-моделирования, хотя для практически используемых его представителей хватает трёх-четырёх уровней.

Так как принятый нами способ описания/моделирования оказался многоуровневым, получающиеся описания (информационные модели мира) тоже будут многоуровневымы. Дальнейшие материалы для чтения по этой тематике можно найти в http://ailev.livejournal.com/1053878.html, а хороший способ проникнуться многоуровневостью “мета” — это посмотреть полутораминутное видео.

### Выбор уровней

Почему же удобно описывать мир именно так, зачем нужна многоуровневость этих "мета"? Оказалось, что они дают возможность поддерживать очень компактное описание мира на нижних уровнях, и обеспечивают связь между множеством таких описаний. Карту с хорошо продуманными обозначениями легко читать, можно быстро получить представление о территории в каком-то аспекте (политическом, климатическом, гидрографическом, …). Но одна и та же легенда может быть использована для построения множества карт разных территорий. А буквы могут быть использованы не только для описания легенды карт, но и для разных других описаний.

Логические уровни оказываются уровнями обобщения мышления для разных дисциплин или даже для групп дисциплин. Если нас интересует мультидисциплинарность (например, мы хотим обсуждать множество инженерных дисциплин, или инженерные и менеджерские дисциплины вместе, или мы хотим включить обсуждение ещё и научных исследований), то нам приходится для качественного обсуждения подниматься выше по лестнице логических уровней.

Важность обобщения мышления состоит в возможности абстрагироваться от содержания нижележащих уровней. Объекты с этих уровней рассматриваются не как полноценные сущности со своими многочисленными особенностями, а как типизированные сущности, с которыми можно выполнять базовые типичные операции. Если мы говорим "знак", то нам всё равно — это греческая буква или точка, знак интеграла или даже пробел. Мы можем говорить о его вхождении в строку, расположении на листе, цвете. Мы абстрагируемся от содержания нижнего логического уровня (какой именно знак) и можем обсуждать операции со "знаками вообще" на более высоком логическом уровне.

Конечно, в разных школах мысли выбирают разные логические уровни. Как и с онтологией, для этого нет заранее известного и истинного решения: оно выбирается исходя из потребностей той или иной деятельности — инженерной, научной, политической. В нашей книге мы будем использовать следующие уровни обсуждения (сверху вниз):

1. Философско-логический: тут надо выбрать предельные онтологии (вопросы "космического масштаба" — есть ли бог, материальна ли вселенная или она представляет из себя симуляцию), знаковые системы, определить их связь с окружающим миром (информационное моделирование). Тут мы договариваемся об общей природе мира, природе описаний, природе интерпретирующего эти описания сознания. На самом деле в нашей книге мы не будем подробно разъяснять предпосылки и рассуждения этого уровня. Работой на этом уровне занимались великие философские логики (Иммануил Кант, Людвиг Витгенштейн, Дэвид Льюис).
2. Формально-математический: тут надо выбрать математический аппарат теории множеств, математической логики, теории графов, теории категорий, или какой-то ещё. При самых солидных философских основаниях необходимо добиваться формальной (математической) непротиворечивости строящихся описаний мира, выражать мысли точно, проверять высказывания на отсутствие формальных ошибок. К этому же уровню относится выбор того или иного языка программирования или языка моделирования. Все современные языки общения с компьютерами имеют под собой твёрдые математические основания, и связаны с соответствующим математическим

аппаратом. Рассуждениями на этом уровне занимаются те же философские логики, но ещё и математики, и физики, и программисты. В нашей деятельности на этом уровне работают специалисты по моделированию данных, computational ontology: они используют математику для того, чтобы единообразно представлять знания в корпоративных менеджерских и инженерных базах данных.

3. Онтологический: надо договориться, что мы видим в мире, какие индивиды, классы и отношения (А ещё мы можем видеть объекты и атрибуты, трансформации, прототипы, или теории — в зависимости от решений, принятых на предыдущих уровнях.) Это уже не предельная онтология с первого уровня, тут нам надо договориться о том, что нужно именно для нашей деятельности. Именно на этом уровне определяется системное мышление, выясняется, в чём суть системного подхода, определяется, что такое "система" или "процесс". Онтологический уровень тесно связан с уровнем математического формализма, но отличается от него. Если в мире есть "системы", то на более высоком уровне можно использовать теории категорий (и записывать знание теоркатегорными уравнениями) или теорию множеств (и записывать логические предикаты). Этот выбор сделают те самые специалисты по моделированию данных, они сумеют совместить в одном вычислении только то, что относится к одной онтологии, к одной непротиворечивой картине мира. Примером совместной работы специалистов одновременно на формально-математическом и онтологическом уровнях являются различные стандарты представления данных, такие, как ISO 15926.
4. Деятельностный: далее нужно научиться описывать человеческую деятельность, описывать дисциплины с их основными понятиями. Для этого нужно задать язык описания практик работы, язык планирования и организации работы, междисциплинарного взаимодействия, работы конкретного предприятия. На этом уровне определяются стандарты ситуационной инженерии методов, архитектуры предприятия, описания бизнес-процессов, органиграмм и т.д. С использованием понятий онтологического уровня на языке математического аппарата описываются стандартные классы и отношения, создаются справочные данные, специфичные для деятельности.
5. Профессиональный: на этом уровне описываются конкретные виды менеджмента, инженерии, научных исследований и прочих профессиональных видов деятельности. Иногда этот уровень называют "методологическим" — на нём описываются методы профессиональной работы, достаточные для решения каких-то узких классов задач. Именно тут задаются основные понятия разных менеджерских и инженерных дисциплин (например, "value proposition" для менеджмента, "архитектура" для системной инженерии, "здание" для гражданского строительства). Занимаются этим профессиональные методологи и "рефлексирующие" инженеры и менеджеры, и делают они это в организациях по стандартизации, профессиональных ассоциациях, университетах. На этом уровне продолжается развитие стандартных справочных данных, уже для отдельных специальностей и профессий.
6. Предпринятия: на этом уровне описывается происходящее в конкретном предпринятии. (Это не опечатка: "предпринятие" — от слова "предпринять",

но вовсе не “предприятие” как юридическое лицо. “Проект” в смысле проектного управления – это тоже ”предпринятие”). Тут с использованием профессиональных понятий моделируется (уже не мета-моделируется!) то, что происходит “в жизни”, в привязке к конкретным условиям работы в ООО “Незабудка” по проекту ремонта лифта "к празднику". Занимаются этими описаниями люди конкретного предпринятия, используя понятия и методы, созданные на всех верхних уровнях.

Повторим, что границы между логическими уровнями расплывчаты, но в каждом конкретном случае мы можем увидеть цепочку мета-моделирования, где тщательно обсуждённые одними профессионалами понятия используются для построения понятий другими профессионалами — от самых абстрактных до самых конкретных.

*Математические формализмы*

Основной объём этой книги будет посвящён рассказу об интересующей нас предметной области на 4-м и 5-м логических уровнях. Эти уровни для нас включают деятельностную парадигму в целом, системное мышление, дисциплины системной инженерии и инженерного менеджмента. Чтобы перейти к этому материалу, ещё немного задержимся на 2-м и 3-м уровнях (формально-математическом и онтологическом), в дополнение к рассказу о 4D экстенсиональной онтологии, с которого мы начинали.

Разумеется, 2-й и 3-й уровни плотно смыкаются с 1-м, философско-логическим. Посмотрите, как обсуждает стык индуистской философской традиции, онтологический статус математики с заходом на лингвистику и гомотопические алгебры математик Роман Михайловский вот тут: http://baaltii1.livejournal.com/573648.html. А вот как обсуждают этот вопрос физики: “вещество происходит из информации”, или “информация происходит из вещества” — http://fqxi.org/community/essay/winners/2013.1

В целом применения математики к реальной жизни сегодня распадаются на два широких класса методов:

- Точные математические методы, воплощённые в точных компьютерных вычислениях (hard computing). К этим методам относятся поиск точных решений уравнений, формальные преобразования, символьные вычисления. Про используемую при этом математику и её влияние на естественные науки есть статья Eugene Wigner "The Unreasonable Effectiveness of Mathematics in the Natural Sciences" — https://www.dartmouth.edu/~matc/MathDrama/reading/Wigner.html
- Приближённые математические методы, воплощённые в алгоритмах так называемых мягких вычислений (soft computing). Сегодня к этому классу относятся чрезвычайно широко используемые нечёткие вычисления, статистические методы, генетические алгоритмы, нейронные сети. Про соответствующую математику есть статья с упором на работу с текстами Alon Halevy, Peter Norvig и Fernando Pereira из Google (2009) "The Unreasonable Effectiveness of Data" — http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/35179.pdf .

При математическом моделировании реального мира принято начинать с моделирования точными методами, в инженерии вообще принято опираться на точные модели. Однако поиск решений для поставленных в ходе моделирования

задач всё чаще осуществляется через тот или иной тип мягких вычислений.

Первое на что надо изучить при выборе того или иного формального математического аппарата – это определённую нотацию (notation). Нотация – это соглашение о знаках, то есть мета-модель того, что мы будем записывать. Наши математические утверждения должны быть записаны так, чтобы человек или компьютер могли их понять. Так же как и в обычной речи, в математике и в программировании одни и те же понятия и связывающее их утверждения можно выразить на разных по виду формальных языках: графических, текстовых для чтения человеком, текстовых для чтения компьютером (бинарных файлах). Языки могут быть основаны на разных разделах математической науки: на теории множеств, логике предикатов, графах и гиперграфах, деревьях и ссылках, текстовых строках и грамматиках, струнных диаграммах и формулах теории категорий, и т.д. Предпочтения тех или иных формальных систем в программировании привели к формированию семейств языков, объединённых определённым языковым стилем (языковых стилях).

Многие формализмы допускают работу с несколькими нотациями, и позволяют перекодировку между ними, например, так можно заменить логические предикаты операциями с графами.

Онтология предприятия из стандарта OMG SBVR, разработанная в логическом формализме, может быть использована в нескольких формальных нотациях: RuleSpeaks (контролируемый естественный язык, ограниченное подмножество английского языка), ORM (Object-Role Modeling, факт-ориентированный язык), UML (Universal Modeling Language, объект-ориентированный язык), Common Logic (язык логики предикатов). Кроме того, стандарт содержит указания на то, как следует выражать эту же онтологию и на других формальных языках.

Из полезных инструментов для формального описания упомянем ещё “псевдокод”. Текст на псевдокоде понятен для человека, и при этом выглядит так, как будто это программа, которую может исполнить и компьютер, однако на самом деле программой не является. Настоящий код на языке программирования может быть выполнен компьютером с использованием формальной семантики этого языка, т.е. точного знания о том, как компьютер должен выполнять вычисления. Псевдокод же только для непосвящённых выглядит как код, у выражений нет формальной семантики, не определён математический смысл используемых понятий. Поэтому компьютер не может исполнить псевдокод, человек-программист должен понять, что описал на псевдокоде автор текста, а потом закодировать это на каком-то языке программирования.

Мы ограничимся рассказом о двух формализмах описания данных, используя при этом формализмы и нотации теории графов, теории множеств и математической логики:

- классического атрибутного описания;
- факт-ориентированного описания, наиболее подходящего для теоретико-множественного подхода.

Кроме того, мы упомянем о наличии теоркатегорных описаний, базирующихся на математической теории категорий (не путать с философскими категориями!). Иные формализмы мы в этой книге обсуждать не будем, но интересующимся вопросом советуем посмотреть ряд обзоров: http://plato.stanford.edu/entries/concepts/ , http://www.iep.utm.edu/concepts/ , и далее попробовать понять, чем занимается

mind and cognitive science (не путать с cognitive psychology, cognitive linguistics!): http://www.iep.utm.edu/category/m-and-e/mind-cog/ . А чтобы понять, насколько формальные подходы далеки от “обыденного” подхода, зафиксированного в бытовом человеческом языке, нужно читать книжку George Lakoff "Women, Fire, and Dangerous Things".

### Объекты и атрибуты

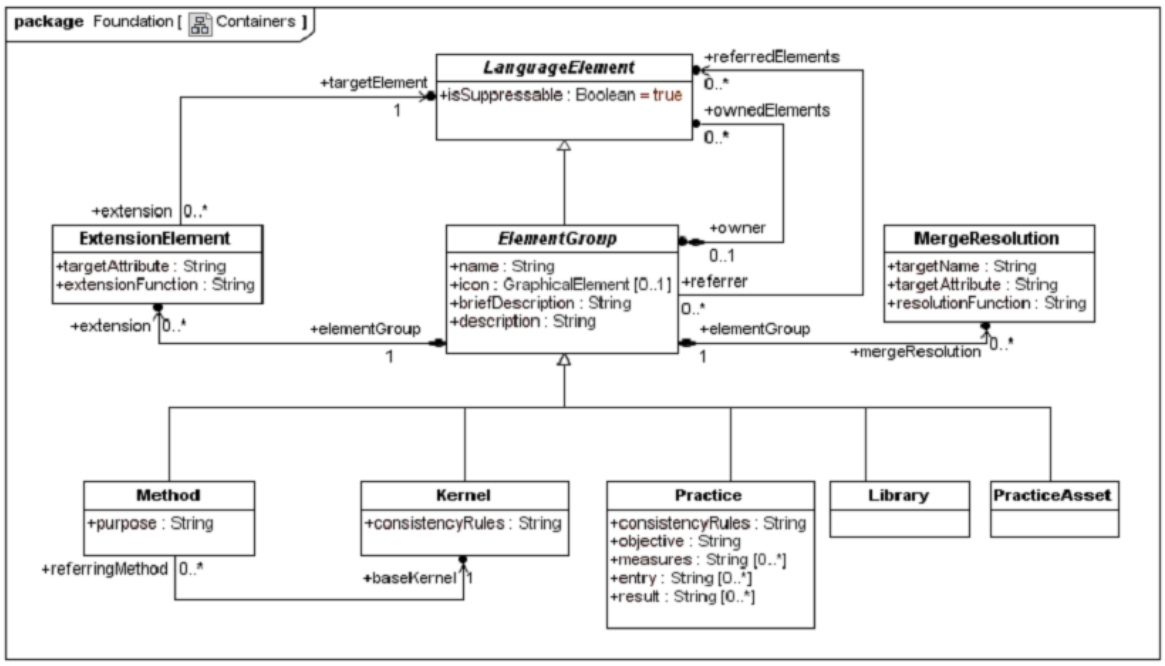
Наиболее распространённым формализмом описания данных является мета-модель “объект-атрибут”. В ней всё в мире представляется как объекты, индивидуальные признаки которых описывают атрибуты. В той или иной форме такая мета-модель используется и в объект-ориентированном программировании, и в языке UML, и в основанном на нём языке моделеориентированной системной инженерии SysML, и в языке мультифизического моделирования Modelica. Но самым известным примером использования объект-атрибутного формализма являются реляционные базы данных. Или их упрощённый вариант – электронные таблицы.

Это формализм является, пожалуй, и самым старым — он ведёт своё начало от работ Аристотеля. А популярен он потому, что простая, ещё не электронная, таблица – это самый очевидный способ систематизировать информацию на плоскости – на листе бумаги, пергамента, и даже папируса.

Каждый класс/тип объектов представляется своей табличкой, со своим набором колонок. Заголовки колонок – это и есть названия атрибутов. Каждая строка – это очередной элемент класса, а каждая клеточка на пересечении колонки и строки содержит значение данного атрибута для данного элемента (число, текст). Значением атрибута может быть и ссылка, чтобы атрибут мог указывать на другие объекты/элементы. Но чаще всего отношения между объектами присутствуют в объект-атрибутной модели непосредственно (реифицированы), и сами, в свою очередь, имеют атрибуты.

Однако именно атрибуты (а не отношения!) являются в объект-атрибутной модели способом выделить объект из класса ему подобных. Отношения могут возникать, изменяться и исчезать (для учёта этого могут использоваться атрибуты отношений), а вот изменения атрибутов объектов мета-моделью, как правило, вообще не предусмотрены. Точнее говоря, в каждый момент времени у объекта есть только одно значение атрибута, и сохранение истории его изменений мета-модель не предполагает

Вот модель объекта Language из стандарта OMG Essence, записанная на языке OMG CMOF, это как раз объект-атрибутный подход.




Прямоугольники на диаграмме – объекты, отношения между ними обозначены линиями с разными типами стрелочек на концах, а атрибуты указаны в прямоугольниках именами с плюсиками перед ними. Объект Method имеет атрибут Purpose (значением которого будет текстовая строка), этот объект является подклассом объекта ElementGroup, у которого много атрибутов — имя, иконка, краткое и полное описания, и все они наследуются, в том числе и объектом Method.

Атрибуты очень понятны и привычны: есть автомобиль, у него есть атрибут "цвет", значение атрибута "цвет" для проезжающего автомобиля — красный. Похожим образом действительно описывал мир Аристотель, а когда были изобретены объект-ориентированное программирование и реляционные базы данных, информатика стала непредставима без атрибутов.

Для атрибутной мета-модели нет простого основания на философско-логическом и онтологическом уровнях. Как говорить об атрибутах без объектов? Что такое "цвет" без объекта? "Красный" — это может быть атрибут объектов машина, лазер, звезда-карлик. Это один и тот же "красный"? Как это обсуждать?

Но и в сугубо практической деятельности возникают трудности. Если вам нужно сложить две объект-ориентированные модели мира, составленные разными людьми – могут возникнуть существенные проблемы. С довольно большой вероятностью то, что отмоделировано в одном проекте как “объект”, будет в другом проекте “атрибутом”. И просто “слить вместе” такие две модели мира в одну будет нельзя, придётся эту модель переработать.

Например, два студента заводят таблички в базах данных двух лабораторий — один табличку лазеров с атрибутами "вес" и "частота", а другой табличку конденсаторов с атрибутами "вес" и "ёмкость". А третий студент спроектировал базу данных склада, в которой табличка называется “оборудование”, и в ней есть атрибуты "вес" и "габарит", плюс атрибут "тип оборудования", с возможными значениями "лазер” и “конденсатор” (и это ещё хороший вариант, потому что атрибут "тип оборудования" может оказаться просто текстовой строкой, в которую рано или поздно запишут и "канденсатор" и "лазир"). Вспомним функциональную нотацию, она неплохо подходит для описания атрибутов (на самом деле это как раз пример

использования псевдокода):

вес: ЛАЗЕР → Real

частота: ЛАЗЕР → Real

вес: КОНДЕНСАТОР → Real

ёмкость: КОНДЕНСАТОР → Real

вес: ОБОРУДОВАНИЕ → Real

габарит: ОБОРУДОВАНИЕ → Real X Real X Real

тип оборудования: ОБОРУДОВАНИЕ → {"лазер", "конденсатор"}

Если четвёртому студенту поручат сделать общую систему автоматизации – он не сможет понять, какая модель мира должна быть: как в лаборатории, где лазеры и конденсаторы объекты, или как на складе, где это просто атрибуты оборудования? Заметим ещё, что в первой базе атрибут "вес" будет иметь имя "Вес", а во второй – "Weight", и заполнен в первой базе будет в килограммах, а во второй – в граммах. Для решения задачи объединения всех баз нарисовать картину мира придётся заново, просто сложить результаты работы первых трёх студентов не получится.

Разумеется, на эти вопросы выработаны ответы, и у философов, и у программистов. Но эти ответы очень непросты. Атрибутный формализм (иногда говорят "парадигмы моделирования") критикуют вовсе не за отсутствие формальности, тут всё в порядке, математическое обоснование у него есть. Критикуют именно неудобство моделирования мира, низкую вероятность того, что два независимых модельера без предварительных консультаций сделают одинаковые модели, пригодные к непосредственному объединению.

### Объекты и факты

В начале 20 века теория множеств получила твёрдые математические основания и стала основой математической логики. Мы уже обсуждали преимущества теоретико-множественного подхода к онтологиям, теперь мы продвинемся в рамках этого подхода чуть дальше.

Основанной на теории множеств альтернативой стал факт-ориентированный способ моделирования мира. В середине 20 века именно этот способ был поддержан аналитической философией. Как говорил Витгенштейн, мир нам дан не "в объектах" — мир нам дан "в фактах об отношениях объектов". Это философская позиция - мы ничего не можем сказать про сами объекты, все наши утверждения делаются про отношения между объектами, объекты проявляют себя в отношениях.

Как представить себе мир, в котором есть только объекты и отношения, а атрибутов у объектов нет? Как представить красный автомобиль? Очень просто: КРАСНЫЙ – это множество всех красных предметов, которые были, есть или будут в мире. Автомобили, лазеры, звёзды, имеющие красный цвет – все вместе и составляют этот класс. А утверждение "мой автомобиль имеет красный цвет" превращается в отношение классификации/членства в классе КРАСНЫЙ.

мой автомобиль є КРАСНЫЙ

Мы помним, что сами классы тоже могут быть классифицированы – принадлежать классификаторам, классам классов. Таким классом классов оказывается ЦВЕТ – это класс, членами которого являются классы КРАСНЫЙ, СИНИЙ, ЗЕЛЁНЫЙ, определённые так, как сказано выше.

ЦВЕТ = { КРАСНЫЙ, СИНИЙ, ЗЕЛЁНЫЙ, … }

Атрибуты в теоретико-множественном подходе стали классами.

Теперь мы можем обсуждать "цвет вообще" и "конкретный цвет" в отрыве от имеющих цвет объектов!

Но и утверждение "любой автомобиль имеет цвет" никуда не потерялось. В атрибутной модели оно выглядело как обязательная колонка "цвет" в таблице автомобилей. В факт-ориентированной модели этому соответствует более сложное утверждение "класс АВТОМОБИЛЬ классифицирован классами класса классов ЦВЕТ". Или, на математическом языке:

$\forall$ xєАВТОМОБИЛЬ $\exists$ yєЦВЕТ (x є y)

Опора на использование отношения классификации уже не позволит легко совершить ошибку, введя текстовый атрибут "тип оборудования" (в рамках такого моделирования это именно ошибка!). Класс ОБОРУДОВАНИЕ включает подклассы ЛАЗЕР и КОНДЕНСАТОР:

ЛАЗЕР $\subset$ ОБОРУДОВАНИЕ

КОНДЕНСАТОР $\subset$ ОБОРУДОВАНИЕ

При этом каждый из них принадлежит ещё и классу всех материальных объектов, то есть обязательно является подклассом одного из классов предметов с данным весом, и одного из классов предметов с заданными высотой, длиной, шириной:

$\forall$ xєМАТЕРИАЛЬНЫЙ ОБЪЕКТ $\exists$ yєВЕС (x є y)

ЛАЗЕР $\subset$ МАТЕРИАЛЬНЫЙ ОБЪЕКТ

ind002233 є ЛАЗЕР

1.25 кг є ВЕС

ind002233 є 1.25 кг

Факт-ориентированное моделирование мира целостно, но так мыслить о мире не очень привычно. Аристотелевский подход царил пару тысяч лет, а теория множеств как средство моделирования мира существует всего-то сто лет!

Современная информатика и инженерия медленно осваивают факт-ориентированное моделирование. Толчок к этому переходу дало решение проблем больших данных (Big Data). Большие данные определяются как такие данные, для которых важны одновременно большой объём, большая вариабельность, большая скорость, большая достоверность (big volume, big variety, big velosity, big veracity). Факт-ориентированность помогает справиться как раз с большой вариабельностью – взаимосвязанные данные приходят из разных источников, собираются там по разным правилам, хранятся в разных базах данных – то есть проблемы объединения разных данных надо решать регулярно и быстро. А как раз при факт-ориентированном моделировании гораздо больше вероятность того, что данные просто сложатся вместе, без существенной перестройки.

Подробнее про факт-ориентированное описание в логической парадигме по сравнению с атрибутным подходом можно прочитать в уже упоминавшейся выше книге Chris Partridge “Business Objects: Re-Engineering for Re-Use”.

Факты и графы

Самым распространённым формализмом для факт-ориентированного моделирования стали ориентированные графы.

Философы и логики предложили простой способ записывать факты, согласующийся с тем, как факты отражаются в человеческой речи и даже в человеческом мышлении. Каждый факт может быть представлен как упорядоченная тройка (triple)

<субъект, предикат, объект> (<subject, predicate, object>).

Например:

<Королева Елизавета, является_матерью, Принц Чарльз>

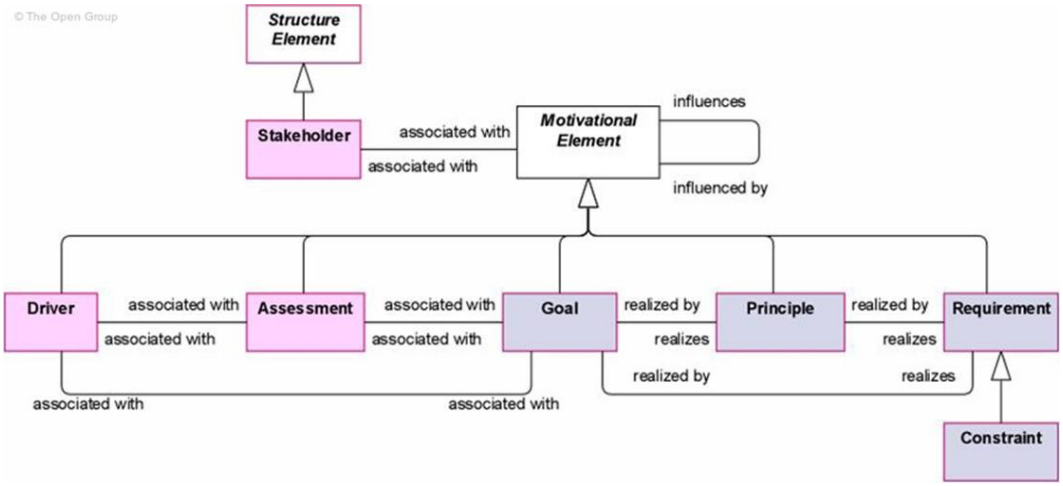
или

<Королева Елизавета, классифицирована_как, МАТЬ>

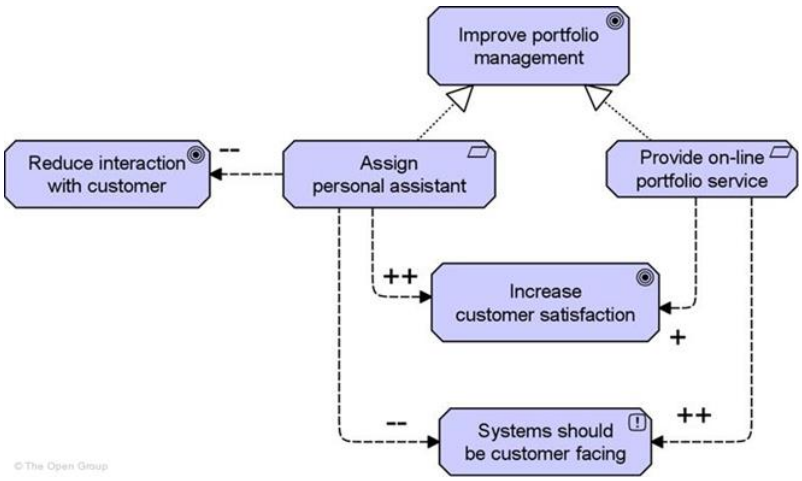
В естественных языках похожей структурой является связка "подлежащее-сказуемое-дополнение".

Упорядоченные тройки являются естественным представлением ориентированного графа с нагруженными рёбрами: субъекты и объекты образуют узлы, стрелки/рёбра/связи направлены от субъектов к объектам, а предикаты - подписи на рёбрах. В таких моделях никаких атрибутов у узлов нет, только связи.

Вот пример факт-ориентированного описания языка архитектуры предприятия OpenGroup Archimate 2.1 (http://pubs.opengroup.org/architecture/archimate2-doc/chap10.html). Мета-модель описания целей и требований для архитектуры предприятия выглядит так:




На следующей диаграмме – соответствующая этой мета-модели конкретная модель (следующий логический уровень):




Разные виды квадратиков и стрелочек — разные типы объектов и отношений. Но никаких атрибутов!

Про связь факт-ориентированных представлений с логикой можно прочесть на http://www.jfsowa.com/ontology/ontometa.htm, подробная информация по математическому аппарату для факт-ориентированных представлений — http://www.jfsowa.com/logic/math.htm .

Факт-ориентированный подход может встретиться вам под разными именами. Наиболее популярным формализмом в этой области сегодня является разработанный для Семантического Интернета (Semantic Web) язык RDF (http://en.wikipedia.org/wiki/Resource_Description_Framework). Множество стандартов используют RDF для факт-ориентированного представления информации, например, ISO 15926 для представления инженерных данных.

В связи с описанными преимуществами растёт применение графовых баз данных, специально предназначенных для хранения данных, не укладывающихся в реляционные таблицы. Графовое представление не так компактно, как атрибутное, но гораздо более универсально.

### Теория категорий

В 21 веке начали обсуждать возможность использовать для представления мира ещё один раздел математики – теорию категорий, раздел высшей алгебры, который "начинается с наблюдения, что многие свойства математических систем можно представить просто и единообразно посредством диаграмм" (стр. 12 из Маклейн С., Категории для работающего математика. М.: Физматлит, 2004). Теория категорий видит в мире объекты, связанные особого рода отношениями – "морфизмами" (преобразованиями).

Теория категорий ещё более непривычна людям, чем теоретико-множественный подход. Но если говорить о компактности представления мира и его универсальности, то теория категорий выглядит более многообещающе, чем теоретико-множественное представление. В настоящее время есть только отдельные результаты применения теоркатегорного аппарата к задачам моделирования в разных предметных областях. Стандартов, регламентирующих использование теории категорий для представления мира, нет.

Мы не будем касаться в нашей книге теории категорий, но советуем хотя бы знать о существовании этого раздела математики. Начальное представление о теор-

категорном анализе диаграмм можно получить серии записей в блоге http://sober-space.livejournal.com/tag/%D0%BA%D1%83%D1%80%D1%81%20%D1%81%D1%82%D1%80%D1%83%D0%BD%D0%BD%D1%8B%D0%B5%20%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B .

Некоторое количество ссылок на приложения теории категорий для компактификации и универсализации онтологического знания можно найти в http://ailev.livejournal.com/1154791.html .

Теория категорий всё более используется в системной инженерии и программной инженерии. Вот материалы докторской диссертации Сергея Ковалёва, в котором можно об этом почитать подробней: http://www.ccas.ru/avtorefe/0001d.pdf

## *Моделеориентированность*

### Что такое модели

Мы довольно часто использовали выше слово "модель". В основном – в сочетании "информационная модель". Давайте обсудим, что же это такое.

Самое общее определение модели — это использование одного объекта (модели) для вынесения суждений о другом (моделируемом) объекте.

Объект M является моделью объекта S только для какого-то определённого лица. Для других лиц тот же самый объект M может восприниматься как вовсе не связанный с объектом S. Объект M может использоваться для получения ответов только на некоторые вопросы в отношении S, или только на протяжении какого-то конечного промежутка времени.

Модели могут быть физическими (макет самолёта для продувки в аэродинамической трубе) или информационными (набор формул на бумаге, файл или целый программный комплекс в системе автоматизации проектирования).

Каково онтологическое отношение между моделью и моделируемым объектом? Стандартного ответа на этот вопрос нет, в разных онтологиях на него отвечают по-разному. Иногда это отношение называют отношением "представления" (representation) – модель представляет объект. Никакой особой теории на счёт отношений representation нет, и надо учитывать, что в некоторых других онтологиях representation называют совсем иное отношение. В некоторых случаях говорят, что модель является описанием (description) моделируемого объекта. Так модно сказать о моделях, которые позволяют вынести суждение о моделируемом объекте только путём их прочтения, без возможности узнать что-то дополнительно к написанному, например, что-то посчитать. Такое моделирование называют структурным, логическим.

Модели, по которым можно проводить вычисления, заставлять их как-то действовать в роли моделируемого объекта – обычно называют симуляционными/имитационными моделями. В них разворачиваются во времени какие-то характеристики моделируемого объекта, решаются уравнения, используются численные методы. Современное моделирование пытается объединить эти виды моделей, предлагая всё более и более мощные и выразительные языки моделирования.

В принципе, любой текст (описание объекта), любая картинка, видео или фотография могут считаться информационной моделью. Но мы среди информационных моделей будем особо выделять машинообрабатываемые модели.

Обратите внимание, не "машиночитаемые", то есть те, которые можно записать на машинные носители, а "машинообрабатываемые", то есть те, с которыми компьютер может проводить хоть какие-то осмысленные операции. Хотя бы отвечать на вопрос "из каких частей состоит моделируемый объект?". Это ещё не имитационная/симуляционная модель, но уже машиночитаемая.

К сожалению, подход 4D экстенсионализма не позволяет удобно рассуждать об общих чертах информационных и физических моделей — в этой парадигме очень жёсткие критерии различение индивидов и абстрактных объектов.

Однако в нашей книге мы до сих пор писали об информационных моделях, и далее будем в основном рассказывать именно о них. Более того, во многих случаях мы будем говорить о моделях данных, то есть моделируемый объект у нас будет данными о каком-то ином объекте, моделью ещё чего-то. Вспомним, что мы называли такую ситуацию мета-моделированием. Для таких мета-моделей - моделей данных – отношение между мета-моделью и моделируемым информационным объектом можно онтологически описать ещё одним понятием – определение (definition). Действительно, в этих случаях мета-модель позволяет оценить правильность, корректность данных, верифицировать их соответствие каким-то требованиям.

### Онтологизирование, моделирование, программирование

Вы можете заподозрить, что описанные нами онтологизирование (составление онтологического описания реальности) и моделирование (создание информационной модели реальности) — это одно и то же. Так считают многие:

“What an engineer calls a model a logician calls an axiom set; what a logician calls a model an engineer calls a simulation. This equivalence of concepts leads to application of well-established methods of logic to engineering."

Henson Graves, http://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:mathemataical_foundation_engineering.pdf

Но отличия, разумеется, есть. Более точно можно сказать, что онтологизирование есть подвид моделирования, подчиняющийся некоторым правилам.

При онтологизировании/онтологическом моделировании считают хорошим тоном, чтобы описания разных предметных областей удовлетворяли хоть какой-то непротиворечивой общей модели мира – предельной онтологии философско-логического уровня.

Однако все логические уровни описания можно считать уровнями моделирования. Просто при моделировании на нижних уровнях работают не столько с *онтологией* (что есть непротиворечивым представлением о мире в целом), сколько с *онтиками* (представлениями о том, какие объекты есть в каком-то маленьком кусочке мира. Две онтики вполне могут давать противоречивые описания, как будто они части совершенно разных онтологий). Онтики и онтологии часто путают, не различают даже в профессиональной речи, но приверженцы классической онтологической традиции всё же учитывают, что онтика пригодна обычно для решения одного класса задач, онтологии же пригодны для решения любых задач, поскольку они содержат универсальные утверждения про весь мир.

Онтологическое моделирование/онтологизирование часто называют “представлением знаний” (knowledge engineering), то есть формализацией знаний в

каких-то структурных/онтологических моделях. Для этого используются формализмы языков представления знаний (Common Logic, OMG SBVR, OWL 2, CYCL), такие языки в большинстве своём основаны на логике предикатов, теоретико-множественной математической парадигме.

Всё чаще и чаще и программирование тоже понимается как моделирование. Языки, разработанные специально для симуляционного/имитационного моделирования просто неотличимы от языков программирования. Первый объект-ориентированный язык программирования (Simula 67, 1967г.) был создан именно как язык моделирования. Современный язык мультифизического моделирования Modelica мало отличается от языка программирования по синтаксису, но компилятор этого языка может собрать разбросанные по тексту программы уравнения, составить из них систему уравнений и решать её алгебраически и/или численно.

### Зачем моделировать

Но зачем нам вообще моделировать?! Зачем что-то "ограниченно замещать", когда можно работать с самим моделируемым объектом, а не моделью и не иметь никаких ограничений?

"Моделирование в широком смысле – это эффективное по затратам использование чего-то одного вместо чего-то другого для мыслительных целей. Это позволяет нам использовать вместо реальности что-то такое, что проще, безопаснее или дешевле чем реальность для заданной цели; модель является абстракцией реальности в том смысле, что она не может представить все аспекты реальности. Это позволяет нам иметь дело с миром упрощённым способом, обходя сложность, опасность и необратимость реальности" (Modeling, in the broadest sense, is the cost -effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality. "The Nature of Modeling.", Jeff Rothenberg, in Artificial Intelligence, Simulation, and Modeling, L.E. William, K.A. Loparo, N.R. Nelson, eds. New York, John Wiley and Sons, Inc., 1989, pp. 75 -92, http://poweredge.stanford.edu/BioinformaticsArchive/PrimarySite/NIHpanelModeling/RothenbergNatureModeling.pdf).

Мы не тратим силы на обсуждение и обработку ненужных деталей моделируемого объекта. Модели это "правильные упрощения".

Формальную (на основе математики) модель можно проверить — вручную или даже компьютером. Это называется model checking. Так, в радиосхеме можно формально удостовериться, что все её компоненты соединены и нет неприсоединённых компонент, все соединения не имеют разрывов (то есть не идут откуда-то в никуда).

Формальную модель можно подвергнуть оптимизации (в том числе компьютерной).

Где физических объект (систему) изготовить долго и дорого, можно ограничиться быстро составляемой информационной моделью, и всё-таки получить ответ на вопрос.

Формальную модель можно не делать руками, а породить (generate) компьютером — это решение проблемы сложности. Так, 7 миллиардов транзисторов в современном микрочипе невозможно нарисовать руками на кремниевой пластине, и даже невозможно руками нарисовать принципиальную схему такого микрочипа.

Но можно породить и принципиальную схему и литографическую маску из моделей более высокого уровня на языке описания микросхем.

Модель позволяет явно увидеть невидимое. Нельзя увидеть в реальной системе мощность или прибыль. Если показать пальцем на двигатель, и спросить "где тут находится его мощность?", то все только посмеются. Если попросить "в каком шкафу компании лежит ваша прибыль — покажите", то все только посмеются. Но и мощность и прибыль можно увидеть в модели.

В итоге с моделями мы работаем быстрее, безопасней, легче и дешевле, чем без моделей — особенно это относится к компьютерным формальным моделям.

В системной инженерии использование моделей настолько важно, что будущая системная инженерия называется Model-Based Systems Engineering (MBSE), прежде всего речь идёт о моделях требований и архитектурных моделях. В менеджменте тоже начало использоваться моделирование: архитектурные модели предприятия, модели бизнеса, модели рабочих процессов, финансовые модели.

Можно ли без моделей? Можно ли обсуждать сложные вопросы так, как это происходит обычно на совещаниях, "с голоса", без моделей? Да, гроссмейстеры продумывают партии "внутри головы", не глядя на доску. Но не все люди — гроссмейстеры, плюс обдуманные в голове партии могут содержать невидимые другим людям ошибки, ибо не обсуждены. Более того, чужие (разработанные гроссмейстерами) партии простые люди просто не будут играть — нет к ним доверия, в критических ситуациях люди предпочтут делать какие-то простые ходы, а не прибегать к разработанным гроссмейстерами непонятным хитростям. Ну, или с гроссмейстерами будут ввязываться в длинные разговоры, чтобы разобраться, что там к чему — а это время.

Анри де Гиус (в книге "Живая компания" он описывает опыт принятия крупных проектных решений в Shell) заметил, что использование сценарного компьютерного моделирования не привело к более качественным решениям. Но принятие решений ускорилось примерно втрое, компания начала реагировать на изменения в окружающей среде и внутри себя втрое быстрее: модели позволяли втрое быстрее договариваться, люди с моделями чувствовали себя более уверенными в принятии решений, чем в случае устных обсуждений и текстовых "докладных записок". То же самое относится к MBSE: вместо недели коллективных медитаций на совещаниях — ночь моделирования и утренняя демонстрация готового решения, при том же качестве результата (этот пример взят из частной переписки системных инженеров). Но в случае очень сложных решений без модели можно вообще не справиться: время обсуждения будет больше, чем будет открыто окно возможностей, и без моделирования можно просто не успеть отреагировать на ситуацию.

### Почему моделирование не повсеместно

Если с моделированием всё так хорошо, то почему оно не повсеместно? Вот основные причины (подробнее — http://ailev.livejournal.com/1056723.html):

1. Полезность моделирования неочевидна. Опыт показывает, что множество решений вполне можно выполнить без моделирования. Никто не может предъявить конкретных цифр, что с моделированием эти проекты были бы выполнены быстрее (решения бы по ним принимались быстрее плюс экономия от меньшего количества ошибок). Ничего удивительного, когда-то и строительных чертежей не было, и моделей сопромата для мостостроителей. Хотя и делали какие-то эскизы и макеты,

которые "моделью" язык не повернётся назвать. Кроме того, использование моделирования связано с затратами (закупка софта моделеров, обучение людей), и на одном проекте эти затраты могут не окупиться. Ах, кроме того, некогда закупать моделеры и отвлекать людей от срочных задач. На втором, третьем, десятом проекте ситуация повторяется, и так годами.

2. Существующий софт моделирования (моделеры) и поддерживаемые им языки моделирования плохи: неточно отражают моделируемые объекты. Например, не отражают основное инженерное понятие — понятие "система". Да, каждые пять-десять лет появляются новые языки моделирования и новое поколение программного обеспечения, но по-настоящему хороших и универсальных до сих пор нет. Без языков же и софта моделировать невозможно.

3. Модели (и результаты моделирования, например, расчёт по модели) оказываются непонятными людям. Решение об использовании моделирования принимают часто не непосредственные пользователи модели, а совсем другие люди — менеджеры и финансисты, которые ничего сами в моделировании не понимают, но знают, что в моделях есть ответ на нужные им вопросы. "Продать" им модели, в которых они не могут найти ответы на свои вопросы очень трудно. Современные моделеры не генерируют "попсовую" инфографику, они часто очень убоги в части дизайна. Поэтому решение об освоении технологии моделирования не принимается: уход денег очевиден, польза неочевидна, результаты непонятны.

4. Модели являются хорошим средством накопления знаний в организации. Если никто не заинтересован в накоплении знаний, то и интерес к моделированию слабый: зачем спрашивать модель, если всегда можно спросить Иван Иваныча? Ах, жаль, что он всегда занят, и его ответ нельзя распечатать и послать по почте, и вообще он пару недель болел в прошлом месяце. Но знания-то в нём есть! Мысль же о том, что эти знания можно отделить от Ивана Ивановича в виде модели — эта мысль обычно не обсуждается, тем более что "искусственного интеллекта всё равно не сделать" и "экономически вы это ваше моделирование всё одно не обоснуете" (что правда).

Тем не менее, если посмотреть на происходящее в менеджменте и инженерии за последние пятнадцать лет, то из состояния "почти нигде, кроме самых развитых компаний" моделирование перешло в состояние "почти везде, кроме самых отсталых компаний".

## Информатика

Граница между онтологизированием, моделированием и программированием очень, очень зыбка и неопределённа. Все они — предмет изучения информатики (не путать с computer science), дисциплины по работе агентов (людей и компьютеров) с текстами и кодами.

Текст — понимается тут как "всё есть текст" ("text" is any object that can be "read," [человеком] whether this object is a work of literature, a street sign, an arrangement of buildings on a city block, or styles of clothing), неформален в части семантики и синтаксиса использованного в нём языка.

Код — формальное в части семантики и синтаксиса использованного языка представление какого-то содержания, независимо от сериализации в строку буковок или оставления в виде какого-нибудь графа, таблиц или триплов.

Особо нужно обсуждать действия в информатике: "акты" (в том числе речевые акты) и "вычисления" (в том числе отработка инструкций компьютером).

Информатика-в-малом — это когда работу с текстами и кодами ведёт один агент (компьютер или человек). Информатика-в-большом — когда в работе с текстами и кодами участвует много агентов.

Дисциплины информатики:

- (философская) логика, объектом практик которой является поиск наиболее компактных описаний для связи текстов и кодов с реальным миром, а также выражения связи с реальным миром формальных и неформальных языков, на которых представлены тексты и коды. Можно считать, что философия языка — это тоже сюда.
- когнитивная наука (cognitive science), объектом практик которой является поиск наиболее компактных описаний для понимания (перевод текстов и кодов во внутреннее представление в голове человека) и писательство (порождение текстов и кодов из внутреннего представления в голове человека).
- лингвистика, объектом практик которой является поиск наиболее компактных описаний кодирования текста и отекстовки кодов.
- компьютерная наука (computer science), объектом практик которой является поиск наиболее компактных описаний перекодирования.

Сюда вполне можно отнести и биосемиотику (в варианте http://galicarnax.livejournal.com/39260.html).

Все эти дисциплины обладают немеренным шовинизмом, то есть пытаются прихватить в свой состав практики смежных дисциплин, поэтому границы между ними весьма расплывчаты. Тем не менее, специалисты одной дисциплины практически не понимают специалистов другой дисциплины, ибо их предметы крайне разнятся, а сообщества почти не пересекаются (подробнее: http://ailev.livejournal.com/1008054.html).

Важно понимать, что текст и код тут понимаются как описывающие что-то (возможно, бывшее, будущее или даже фантастическое и невозможное) в мире, то есть интересует даже не просто “математика работы с текстами/кодами”, а и отношение результатов этой работы к окружающему миру — онтологизирование и моделирование акцентируют именно этот аспект, в то время как программирование уделяет этому аспекту недостаточное внимание.

### Принципы моделеориентированности

Работы группы AtlanMod (http://www.emn.fr/z-info/atlanmod/index.php/Main_Page) посвящены подходу model-driven engineering (моделеориентированная инженерия). Как -based, так и -driven оба переводятся как -ориентированная, но имеют разный смысл. -Based означает, что для решения каких-то задач выполняется моделирование. Далее человек смотрит на результаты моделирования (расчёт, текст, чертёж, диаграмму) и делает очередную модель (расчёт, текст, чертёж, диаграмму). А вот -driven означает, что модель является основным, что работает — на неё не “глядят для осмысления, и сочиняют другую модель”, а она компьютерно преобразуется с добавлением новых данных в другую необходимую модель. Часто про model-driven подход говорят как о преобразованиях (transformation) моделей.

Вот десять принципов model-driven engineering, сформулированные Jean Bezivin (один из бывших руководителей группы AtlanMod, далее мы цитируем по его презентации

http://cbi2014.unige.ch/documents/CBI2014.TowardsCrossDisciplinaryPractices.JeanBezivin.pdf)

1. Принцип представления. Любая модель M представляет объект S (в оригинале используется слово "система", но в нашей книжке термин система будет в следующих разделах закреплён только за физическими системами-индивидами, а моделировать можно всё что угодно, в том числе другие информационные модели, т.е. не "истинные системы").
2. Принцип множества групп описаний (veiw). Объект S может быть представлен несколькими моделями. Мы в нашей книге будем это разбирать подробней в последующих разделах и называть мульти-моделированием.
3. Принцип соответствия: любая модель соответствует мета-модели (помним про "логические уровни": моделирование одного логического уровня ведётся с использованием знаний, определённых на другом логическом уровне).
4. Трёхуровневая организация (любая метамодель MM соответствует общей для них метамодели MMM). Впрочем, это Jean Besivin и AtlanMod настаивают именно на 3 уровнях моделирования, в общем случае логических уровней может быть много.
5. Принцип трансформации: наиболее важная операция, применимая к модели — это трансформация. Помним, что группа AtlanMod занимается главным образом переходом от Model-Based engineering к Model-Driven Engineering: их волнует порождение/генерирование по моделям инженерных рабочих продуктов, а не, например, имитационное моделирование или использование моделей для налаживания взаимопонимания между менеджерами, инженерами, клиентами.
6. Принцип HOT (High-Order Transformation): трансформация (код, описывающий преобразование модели) это тоже модель.
7. Принцип переплетения (weaving): отношения между несколькими моделями могут быть представлены тоже как модели.
8. Мегамодель: важна модель, чьи элементы модели и мета-модели плюс отношения между моделями. В мега-модели мы представляем информацию нескольких логических уровней.
9. Принцип унификации. Все упомянутые модели специализируют общую абстрактную (математическую, алгебраическую) модель — высший уровень мета-мета-моделирования.
10. Подход технического пространства (Technical Space Framework): любая модель имеет внешнее техническое представление (больше нет ограничений на выбор одного языка моделирования для какого-то вида моделей: нотации могут быть разными, модель одна).

Проверьте себя, различаете ли вы, когда говорят о мульти-моделировании, мета-моделировании, мега-моделировании.

## 3. Инженерия и наука

Системноинженерное мышление и деятельность — как нам их описать?! Их ведь нельзя пощупать, нельзя наглядно продемонстрировать. Если попробовать пронаблюдать, что делает инженер — он спокойно сидит на стуле и кликает время от времени кнопками мышки. Или спокойно стоит и смотрит на приборы сложной

установки, иногда крутит какие-то ручки и нажимает какие-то кнопки. И в то же время понятно, что его мышление и деятельность отличаются от мышления и деятельности пианиста, который тоже спокойно сидит на стуле и кликает время от времени кнопками рояля.

Наша книга о способах мышления и способах действия. Для начала нам нужно найти способы, которыми мы будем описывать мышление и действие.

Инженерия занимается изменением реальности — двигает горы, создаёт спички и зажигалки, строит марсоходы и медицинские роботы Да Винчи. Пространство-время физического мира были устроены одним способом, пришли инженеры, пространство-время стало устроено по-другому.

Производством компактных описаний реальности занимается наука — придумывает диаграммы Фейнмана, теории мышления, понятия системы и деятельности. Пространство-время думали, что устроено одним способом, пришли учёные, пространство-время теперь думают, что устроено другим способом.

Обучить науке — это обучить тому, как строить компактные и понятные людям описания того, как устроен мир. Например, 4 уравнения Максвелла описывают все электромагнитные явления мира, а уравнение Шрёдингера описывает волновые функции элементарных частиц. Можно ли “выучить на учёного” — это отдельный вопрос, мы его тут не будем рассматривать.

Но для начала нам придётся заняться инженерной наукой: попытками компактно описать то, что делают лучшие инженеры мира.

Тут нужно учитывать, что само понимание того, что такое наука, исследования, инженерия непрерывно меняется, чуть ли не каждый десяток лет. Интересующиеся могут ориентироваться на следующую последовательность ключевых работ по истории и философии науки (подборка Антона Николаенко, https://www.facebook.com/nikolaenko.anton.9/posts/761402053939232):

в 1934 Поппер выпускает свою "Логику научного исследования",
в 1935 Флек публикует "Возникновение и развитие научного факта",
в 1937 Мертон - работу "Наука и социальный порядок",
в 1946 Полани - книгу "Наука, вера и общество",
в 1962 Кун - свою "Структуру научных революций",
в 1971 Бен-Дэвид - свою "Роль ученого в обществе",
в 1972 Тулмин - работу "Человеческое понимание",
в 1975 Фейерабенд - свою книгу "Против метода",
в 1976 Блур - свою "сильную" программу - книгу "Знание и социальное представление",
в 1978 Лакатос - свою "Методологию научно-исследовательских программ",
в 1978 Хюбнер - свою "Критику научного разума",
в 1979 Латур - свою "Лабораторную жизнь",
в 1979 Малкей - свою работу "Наука и социология знания",
в 1998 Шейпин - свою "Научную революцию",
в 1998 Коллинз - свою "Социологию философий",
в 2008 Деар - свою книгу «Событие революции в науке".

Всё из перечисленного переведено целиком на русский язык.

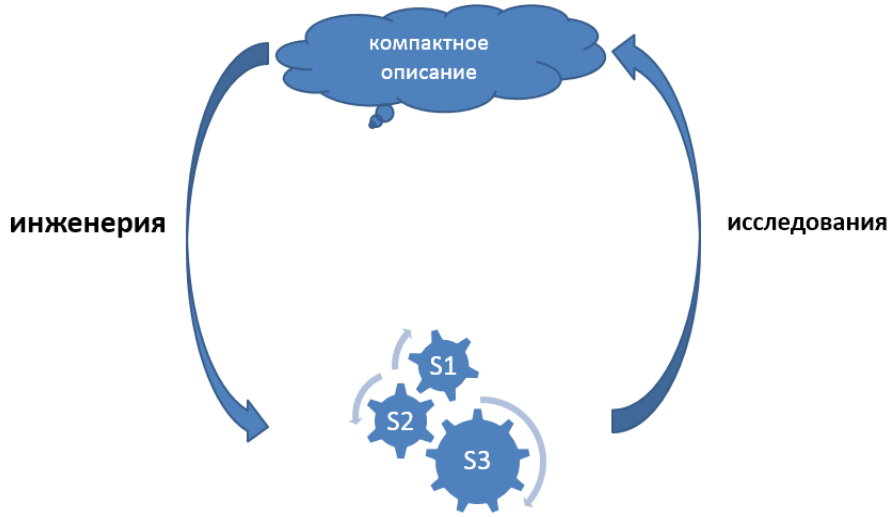
*Инженерия не научна*

Разница между инженерами и учёными

Нельзя путать инженеров и учёных. Учёные ровно обратны инженерам: если инженеры делают реальные материальные вещи, опираясь на мысль, то учёные делают мысли из реальности — получают компактные, понятные и формальные описания действительности. Конечно, инженерия и исследования тесно связаны:

- когда инженер получает инженерный объект, поведение которого не отвечает его замыслу, он исследует проблему: пытается найти наиболее компактное описание работы этой инженерной системы, из которого было бы понятно, в чём он ошибся при замысле и его воплощении. Затем исправляет ошибку: меняет систему.
- когда учёный придумывает новый способ описания, более компактный и лучше объясняющий мир, чем предыдущие способы, то он проводит эксперименты. Отнюдь не все эксперименты "мысленные". Некоторые из них требуют создания весьма и весьма сложных инженерных объектов (например, ускорители — это одни из самых сложных на Земле инженерных объектов). Когда эксперимент проведён, учёные корректируют свои теории в зависимости от результатов эксперимента.

Тем самым инженерная и исследовательская деятельности оказываются связаны в цикл, и в каждом исследовательском или инженерном проекте обычно приходится много раз повторять цикл:




Главное тут — цель объемлющей деятельности, а не собственно сама работа "исследований" или "разработки" ("науки" или "инженерии"): целью является либо появление какой-то материальной системы, приносящей пользу пользователям, либо появление какого-то компактного описания/объяснения того, как устроен мир. В любом случае, либо инженерия оказывается спрятана в исследованиях, либо исследования спрятаны в инженерии.

Эта путаница, отражающая связь инженерии и науки, весьма распространена:

- В большинстве мультфильмов "учёный" в белом халате меньше всего учёный, это "инженер-изобретатель". Эти якобы "учёные" не ставят эксперименты: они придумывают какие-то необходимые для их целей системы, и воплощают эти системы в реальности. Это "лаборатории Эдисона".

- "Прикладная наука" (applied research, прикладные исследования) тоже меньше всего наука, несмотря на слово "исследования". Никаких "теорий" от этой науки ожидать не приходится, а результаты НИОКР (научных и опытно-конструкторских разработок) как правило — вполне себе инженерные объекты, а не способы описаний. Единственная их разница с результатами классической инженерной разработки, так это то, что результатом является прототип или опытный образец, а не идущий потребителю продукт (между "работающим прототипом" и "выпускаемым продуктом" могут быть годы и годы разработки, development — и результатом являются не "изобретения", а "инновации", определяемые как успешно выведенные на рынок изобретения).
- Очень часто прикладные исследования (research) и классическую разработку вообще не разделяют, отсюда устойчивое сокращение R&D (research and development). Есть ли разница? Есть: "лаборатория Эдисона" (лаборатория Bell Labs, лаборатория IBM и т.д.) всё-таки отличаются существенно по организации труда и проходящим в них работам от классической инженерной разработки. Но они не отличаются принципиально! Менеджментом R&D как раз и занимается technology and innovation management.

Настоящая наука — это "basic research", которые ведутся в "лабораториях Эйнштейна". На выходе не "опытные образцы" (inventions, "изобретения", прототипы систем и идеи для этих прототипов), а теории — компактные и формальные описания природы. В отличие от R&D менеджмент и финансирование науки происходят совершенно другим образом, часто они происходят вообще вне рамок предприятий.

Системная инженерия — это тоже инженерия, не наука. Системная инженерия вполне может включать R&D, изобретения, создание прототипов.

Ещё одна связь науки и инженерии — это конструирование экспериментальных установок, в какой-то мере создание инженерных прототипов может быть отнесено к научным исследованиям (хотя речь может идти не о подтверждении научной теории, а подтверждении какой-то догадки или замеченной эвристики).

Если мы хотим создать формальное компактное описание самой системной инженерии — то это наука, "инженерная наука" (engineering science).

### Предмет инженерии и научные предметы для инженерных объектов

Нужно различать предмет самой инженерии (engineering) и предметы, изучающие объекты инженерии — механику, электрику, компьютерную науку и т.д. Инженерия (в том числе системная инженерия) описывает то, как работают инженеры. Инженерная наука (engineering science) даёт описание того, что делают люди в инженерном проекте. Другие предметы и другие науки описывают поведение инженерных объектов, то, как работают они: механические устройства, электрические схемы, компьютерные программы.

Наиболее просто понять разницу между такой "наукой про инженерный объект" и самой инженерией можно на примере computer science (компьютерной науки) и software engineering (программной инженерии). В блоге http://gaperton.livejournal.com/ была замечательная история, как его автор, отличник по computer science, пошёл после ВУЗа покорять иностранную софтверную компанию. Но там ему буквально за два дня объяснили, что он (может быть) умеет программировать, но совершенно точно не умеет работать: нужно было

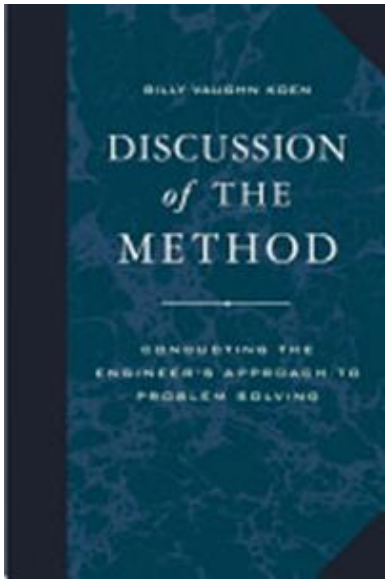
разбираться с тем, как составляются требования, писать нужно было не столько какой-то хитрый алгоритм, сколько заплатку к уже написанному коду на миллион строк, и нужное место для заплатки в этом коде нужно было как-то найти, кроме этого нужно было разобраться в управлении конфигурацией (версионированием), системой управления изменениями, наладить отношения с менеджерами и тестировщиками, понять суть принятой в проекте методологии разработки и т.д. — при этом ничему этому в ВУЗе не учили. Работа оказалась совсем не тем, чему учили: огромное количество времени уходит на общение с коллегами, а не на "думание", время решения задачи не "сколько нужно", а "сколько отведено графиком". Учили computer science (ключевые слова: алгоритм, оценка скорости выполнения алгоритма, нотация, грамматика, реляционная модель данных) и не учили software engineering (ключевые слова: требования, архитектура, тестирование, изменения, сопровождение). Точно так же инженера-механика могут научить работать с прочностью, формой, скоростями, трением с использованием всяческих теорий, но могут не научить работать с требованиями, архитектурой, испытаниями и инженерными обоснованиями. Именно поэтому хороший физик может быть хорошим специалистом по сопротивлению материалов, но не быть хорошим инженером — у него знания про инженерные объекты, но не про саму инженерию.

Обучить инженерному делу как таковому — это обучить тому, как устроен ход инженерной разработки, какие основные понятия предметной области самой инженерии (а не предметной области, описывающей физику и алгоритмику создаваемого инженерного объекта): как устроен инженерный проект в целом и как разворачивается во времени проектирование/конструирование/программирование, изготовление и разворачивание целевой системы, а не только как устроена создаваемая система.

### Ненаучность инженерии. Эвристики

Знания самой инженерии (как что-то сделать) и знания об инженерных объектах (установках, сооружениях, транспортных средствах, компьютерах и т.д.) могут быть как научными, так и ненаучными. Инженерия рождается и живёт методом проб и ошибок, её знания, передающиеся из проекта в проект про неё саму (как что-то сделать) и про её инженерные объекты вовсе необязательно "научны". Так что определения инженерии как "принесение научных знаний в практику" просто неверно, хотя наука и используется там, где она есть.

В книге Discussion of the Method исследователь инженерии Billy Koen приводил пример: если бы в средние века к инженеру пришли с просьбой построить мост, а он бы отказался на основании того, что сопромат изобретут только через 200 лет — что можно сказать о таком инженере? Или если бы современный инженер при просьбе построить ракету, летящую на Луну или Марс, отказался бы от проекта на основании того, что достаточно подробной "теории Луны" или "теории Марса" ещё не создано? Или при необходимости создать робота инженер вдруг говорит, что "теории о том, что делать, чтобы создавать робота пока нет — поэтому я не знаю, что в каком порядке делать, поэтому подождём до тех пор, пока у учёных не появится соответствующего раздела инженерной теории". Отказы на таких основаниях не свойственны инженерам, их не останавливает отсутствие научного знания в том, чем они занимаются.




Billy Koen (http://www.me.utexas.edu/directory/faculty/koen/billy/41/)

Инженерия кроме научных теорий активно использует эвристики (heuristics) — это догадки о закономерностях, которые вовсе необязательно "научны" в традиционном смысле этого слова, т.е. это не "фальсифицируемая теория" по Попперу: инженер с самого начала знает, что эвристики вполне могут быть в его случае ошибочны и неприменимы. Плюс обратите внимание, что инженерные эвристики определяются отличным образом от философской "эвристики", найдите соответствующую литературу сами). Вот примеры инженерных эвристик:

Из Turton, Richard, et al. (2003) *Analysis, Synthesis, and Design of Chemical Processes*, Upper Saddle River, NJ: Prentice Hall.:

- Используй вертикальный резервуар на опорах, когда его объем меньше 3.8м3
- Используй горизонтальный резервуар на бетонных опорах, когда его объем между 3.8 и 38м3
- Используй вертикальный резервуар на бетонных опорах, когда его объем больше 38м3

Большая подборка подобных эвристик для инженерии химических производств дана тут: http://people.clarkson.edu/~wwilcox/Design/heurist.pdf

На эту тему есть хороший анекдот:

> Физику, математику и инженеру дали задание найти объём красного резинового мячика.
>
> Физик погрузил мяч в стакан с водой и измерил объём вытесненной жидкости.
>
> Математик измерил диаметр мяча и рассчитал тройной интеграл.
>
> Инженер достал из стола свою "Таблицу объёмов красных резиновых мячей" и нашёл нужное значение.

Есть и другие типы инженерных эвристик, совсем не связанных с инженерными расчётами и приёмами конструирования-проектирования:

- Стейкхолдеров всегда на один больше, чем вы знаете; известные вам стейкхолдеры всегда имеют потребность (need) на одну больше, чем вам известно (это шестая из основных инженерных эвристик Tom Glib,

http://www.gilb.com/dl97).

- Порядок бьёт класс (в больших проектах упорядоченная работа команды заурядных специалистов бьёт беспорядочную работу высококлассных звёзд).

Тем не менее наука и инженерия тесно связаны: эвристики в более простых системах заменяются научными теориями, в том числе в виде компьютерных моделей (разница: правильно применённая теория даёт надёжный ответ, а эвристика, возможно, врёт), а место для метода проб и ошибок смещается в сторону более сложных систем, которые плохо описываются наличными научными теориями.

Формальные (теоретические, следующие законам логики, а не чисто эвристические — хотя и те и другие могут быть подтверждены экспериментами) описания инженерных систем позволяют проводить формальный анализ: находить ошибки без создания системы, а часто и вычислять необходимые или оптимальные характеристики системы. Очень дорогой метод проб и ошибок с его бесконечным циклом догадок и экспериментов при помощи формальных описаний превращается в совсем другой метод работы, число проб становится в разы и разы меньше. Источником же полезных формализмов (методов описаний самых разных феноменов) является как раз наука. Число формализов растёт, число найденных эвристик тоже растёт, поэтому со временем растёт и уровень инженерии.

В связи с этим любые достижения в инженерии по предложению Billy Koen нужно оценивать не по абсолютной шкале, а на конкретный момент времени, в соответствии с накопленным на этот момент объёмом научного и эвристического инженерного знания — и это "текущее состояние инженерии" Billy Koen предложил называть SoTA (state-of-the-art). Инженерный проект плох, ежели он не использует всей полноты научного и эвристического знания, накопленного на конкретный момент времени. Со временем объем знаний растёт, и инженерные проекты становятся более и более сложными, достигая невозможных для предыдущего времени характеристик.

### Наука как "научение птиц полёту"

Существует мнение, что наука для практической деятельности бесполезна. Практики добиваются успеха не на основе научных знаний, а на основе "возни" (tinkering, ср. "Ну is tinkering with a car" — "он возится с автомобилем").

Эта точка зрения была развёрнута в книге Насима Талеба "Антихрупкость". Он сравнивает учёных с теми, кто приходит к птицам и пытается научить их летать, давая знания по аэродинамике. Типичное высказывание в его книге на эту тему: "Никто не опасается, что ребенок, понятия не имеющий о разных теоремах из области аэродинамики и не способный решить уравнение движения, не сможет ездить на велосипеде". Он защищает метод проб и ошибок, защищает эвристики. Он абсолютно прав. И он прав, когда пишет о создании реактивных двигателей: сначала было много проб и ошибок, потом только появилась теория, а не наоборот.

Тем не менее, исследования дают нам способ думать по-новому: осознанней, быстрее и надёжнее. Но не так, чтобы исследования вообще позволяли нам думать. Думаем мы и без них, но спонтанно, медленно и не слишком надёжно. Метод проб и ошибок всем хорош, кроме того что чрезвычайно дорог и долог. Если есть способ что-то физическое коротко описать, а потом работать с этим описанием-моделью, а не с самим физическим объектом, то так и нужно делать. Если вы учите ребёнка ездить на велосипеде, то вам особой науки не нужно. А если вы учите компьютер

быть автопилотом на реактивном самолёте, то незамутнённым наукой методом проб и ошибок вы загубите слишком много реактивных самолётов.

Ещё один аргумент в пользу науки и компактных описаний появляется, когда вы замечаете, что в книжках Талеба ничего не говорится о коллективной работе. Когда речь идёт не о рынке, где "дальнее взаимодействие" (никто друг друга не знает, сделки между незнакомыми людьми) и где хорошо работают рассуждения Талеба, а когда мы хотим хорошее мышление передать кому-то другому, но не понимаем, из чего это мышление состоит, что передавать. Охота и собирательство талантливых людей хороши, но переход к осёдлому земледелию даёт скачок в производительности труда — выращивать талантов дешевле, чем их выискивать.

Так что сначала нам нужна какая-то наука, чтобы инженерные знания компактно описать — и уже после этого мы их можем передать.

Ещё один аспект инженерной работы — она не делается одиночками. Нужна координация усилий сотен, тысяч и даже десятков тысяч людей. Все эти люди должны как-то договариваться между собой. Как они могут договориться, если каждый про свою часть дела может рассказать примерно столько же, сколько едущий на велосипеде мальчик про свою езду "я чувствую, что я держу равновесие и я чувствую, что на большой скорости надо бы пригнуться"? Компактные описания нам нужны, чтобы люди могли иметь одинаковое описание того, что они делают, чтобы не возникло проблемы строительства Вавилонской башни.

Излагаемый в нашей книге подход к системноинженерному мышлению и действию совершенно необязателен для инженеров-одиночек, среди одиночек всегда найдётся Кулибин или Левша. Но вот если речь пойдёт о какой-то более-менее масштабной коллективной инженерной деятельности, то синхронизация способов обсуждения проекта может сэкономить много-много времени — все ведь помнят проблему, возникшую при строительстве Вавилонской башни? Мы должны научиться описывать другим людям, что мы делаем и почему, чтобы другие люди могли к нам присоединиться.

Конечно, уметь что-то описывать и в инженерии, и в менеджменте (и даже в литературе) вовсе не означает то, что вы опишете что-то ценное и важное. Графоманам никогда не получить Букеровскую премию, хотя они умеют писать. Научиться думать об архитектуре или проектном предложении, научиться компактно "по науке" записывать свои мысли вовсе не означает, что вы что-то придумаете интересное. "Думать и придумать" в этом плане похожи на "учить и выучить", "делать и сделать" — процесс ничто, результат всё. Но если не думать, то и не придумаешь. Если не учить, то и не выучишь. Если не делать, то и не сделаешь. Процесс важен, без него не будет результата.

Так что для начала нам нужна инженерная наука (engineering science), хотя мы точно знаем, что инженерия ("инжиниринг", как сейчас всё чаще говорят) — это не наука. Но нам нужны компактные описания инженерии и менеджмента как минимум для того, чтобы договариваться об инженерии и менеджменте с другими людьми.

И нам нужна наука о мышлении, хотя мы точно знаем, что само мышление — это не наука. Но нам нужны компактные описания мышления, чтобы договариваться о них с другими людьми, чтобы реализовать коллективное мышление. Построить такую ракету, чтобы она долетела до Марса или даже крошечной по космическим меркам кометы — для этого метода проб и ошибок явно недостаточно, но системные инженеры строить такие ракеты научились.

*Инженерия научна*

Инженерная наука

Можно думать о науке, ищущей компактное описание самой инженерии (engineering science) — эта наука должна рассказать о том, что делает системный инженер в ходе инженерного проекта.

Отношение к системной инженерии как к науке среди системных инженеров обычно отрицательное (хотя они и признают её полезность). Это отношение связано с тем, что "наука" обычно никогда не рассказывает, как породить ту или иную идею. Без "науки" трудно решать практические задачи, но сама наука обычно ничего не говорит о том, как породить идею решения. Наука лишь рассказывает о том, какие есть объекты, их характеристики и отношения друг с другом и что можно с ними делать. Что с этими объектами делать для решения конкретных задач — неведомо. Никакое знание физики не помогает решать даже олимпиадные задачки, что уж говорить о решении реальных проблем. Но без знания физики и олимпиадные задачки и инженерные проблемы решать труднее. Наука аналитична, инженерия же связана с решением проблем (проблема — когда никто не знает, что делать) и требует синтеза.

К сожалению, сведение всей инженерии только к науке и "аналитике" в ущерб практической деятельности по решению проблем, в том числе с помощью "ненаучных" по своей природе догадок-эвристик отражено и в учебных курсах по инженерии. Так, руководитель NASA Michael Griffin в речи 2007 г. System Engineering and the Two Cultures of Engineering (http://www.spaceref.com/news/viewsr.html?pid=23775) пишет, что:

> I have always loved the view of the engineering profession captured by the great Theodore von Karman when he said, "Scientists study the world as it is; engineers create the world that has never been." Less eloquently, engineers are designers; they synthesize knowledge to produce new artifacts. Von Karman speaks to what most of us, and certainly most laymen, would consider the essence of engineering: engineers create things to solve problems.
>
> But all of us who are engineers know that the engineering profession also has a rich scientific side, the analysis of these artifacts and the prediction of their behavior under various environmental and operational conditions. Adapting von Karman's observations, it may be said that engineering science is the study of that part of the world which has been created by man.
>
> Sadly, many students have been led to believe that engineering science is engineering! In a curriculum of 120 or more credits leading to a bachelor's degree in a branch of engineering, the typical student is required to take one, or maybe two, courses in design. Everything else, aside from general-education requirements, focuses on the analysis, rather than the creation, of engineered objects. Graduate education often has no design orientation at all. So, engineering as taught really deals with only a part of engineering as it is practiced.

Основное, что должен уметь делать системный инженер — это создавать материальные объекты, а не анализировать их. Упор на синтез, а не на анализ характеризует инженера.

Интересно, что примерно такое же рассуждение проводится про образование менеджеров-предпринимателей. Henry Minzberg считает, что выпускников MBA (Master of Business Administration) готовят как аналитиков, а не синтетиков. Из них получаются не столько предприниматели-руководители предприятий, сколько начальники финансовых и прочих аналитических отделов компаний — тех подразделений, где основным продуктом являются "отчёты". А потом совсем другие люди (часто с инженерным образованием) по этим отчётам принимают синтетические решения и реализуют какие-то идеи на практике.

Один из системных инженеров NASA привёл пример, в котором сравнивает инженерную науку (в том числе ту, которой обучают в университетах) с порнографией: можно сколько угодно тратить времени на просмотр и обобщение бесконечного числа порнофильмов, но все эти часы и часы псевдоопыта "изучения" можно было бы потратить на получение собственного опыта, получив при этом не меньше удовольствия, чем от просмотра порно. Ну, и овладение всеми возможными классификациями увиденного, знакомство с хитрой терминологией и прочие "атрибуты научности", почерпнутые из просмотра видео могут абсолютно не помочь (а то и помешать) в конкретной жизненной ситуации. Так что системные инженеры предлагают больше уделять внимания работе в проектах, а не заниматься чтением учебников.

Системные инженеры постулируют примат инженерного опыта над инженерной наукой, хотя также и признают важность науки (ибо без признания важности науки иначе можно скатиться назад, к древним временам инженерии исключительно проб и ошибок) — просто оформляемый эвристиками инженерный опыт много, много больше той территории, которая уже отвоёвана наукой.

Наша книга решает вопрос о балансе инженерной науки и инженерной практики так:

- Она сначала сосредотачивается о самых общих теоретических знаниях об инженерной науке. Знание принципов освобождает от знания множества фактов. Нужно знать, какие основные объекты, с которыми работают инженеры, в чём суть их работы.
- Она содержит ссылки на материалы о практиках моделеориентированной системной инженерии. Знакомство с конкретными практиками крайне важно, но оно происходит на втором шаге: нужно понимать, что мастерство во владении отдельными практиками набирается достаточно долго, и по каждой практике есть книжки даже более толстые, чем наша.

Теоретическое знание нашей книги должно "оживляться" на материале конкретного инженерного проекта. Книга не рассчитана только на прочтение, она должна лечь в основу практической работы (учебной или реальной), в которой теоретическое знание книги соединяется с практическим опытом его использования. Нет ничего практичней, чем хорошая теория, но без занятий реальной инженерной деятельностью эту практичность предъявить нельзя. Тот, кто прочёл много книг по танцам, но никогда не танцевал сам, вряд ли сможет станцевать при случае. К инженерам (менеджерам, учёным, кому угодно) это тоже относится.

### Научное (формальное) основание системной инженерии

Системная инженерия и менеджмент опираются не на теории, а на эвристики. Они эмпирические, а не "строго научные" дисциплины типа физики или химии.

И тем не менее, есть ли какое-то научное (формальное, теоретическое) основание

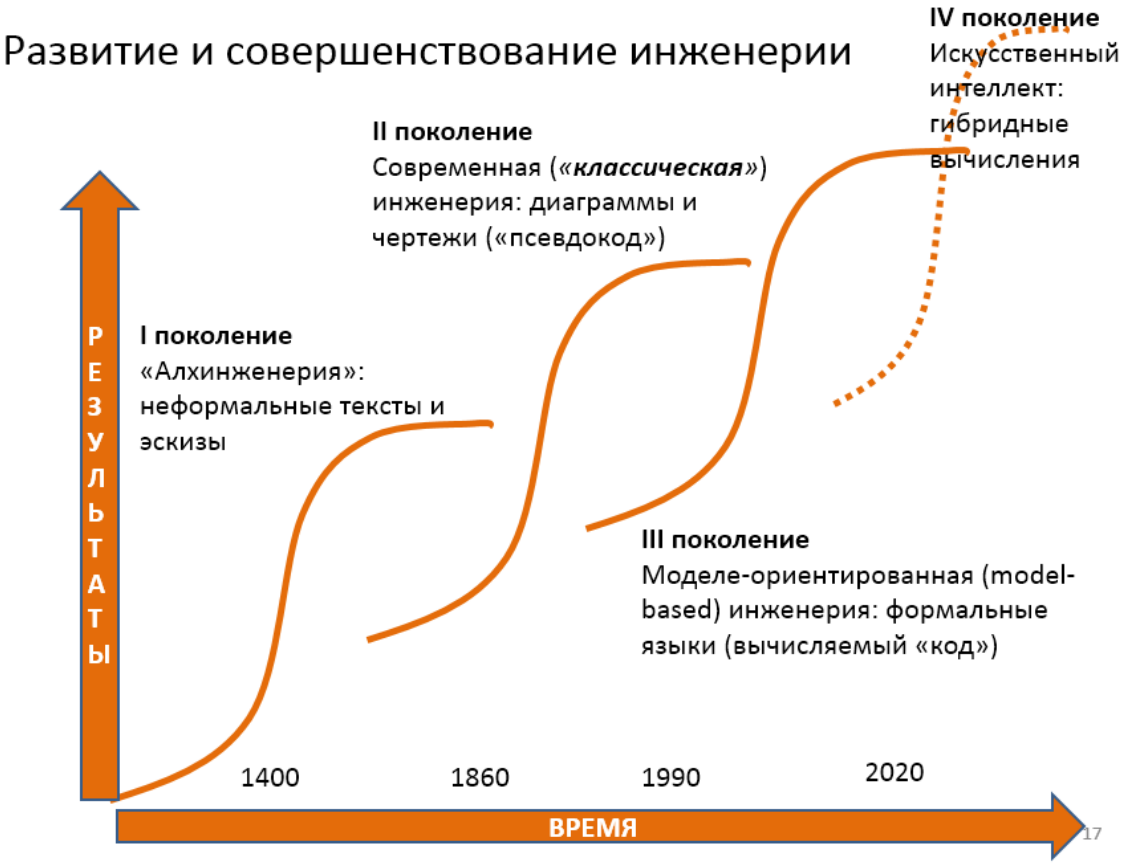
системной инженерии, т.е. существует ли компактное формализованное описание системноинженерного знания? Существует ли оно, или речь идёт только об искусстве (humanities, куда относятся литература, изобразительное искусство, кино и т.д.)? Почему системным инженерам выдают master of science, а не master of art in Systems Engineering?

Действительно, в большинстве случаев системных инженеров учат работать примерно так, как учат писателей, композиторов и прочих людей "творческих профессий": им дают ознакомиться с работами предшественников, рассказывают о том, что в этих работах хорошего и что плохого, как называются те или иные объекты и практик (например, что обычный текст называется "проза", в песне можно выделить "мелодию") учат каким-то отдельным приёмам работы, которые предположительно должны давать хорошие результаты.

Если бы у системной инженерии было научное основание, она была бы полноценной научной дисциплиной, это бы означало, что у неё есть основные понятия-"идеальные объекты", в терминах которых описывается окружающий мир — примерно так, как у физики есть понятия физического тела и поля, а у химии есть понятие химической связи. Конечно, в системной инженерии ключевым понятием является "система", но использование этого понятия осложняется тем, что оно пока онтологически (философско-логически) недоопределено, и прямо сейчас идёт интенсивная работа по его формированию. Главным образом эта научная (точнее, научно-методологическая — созданием новых дисциплин занимается методология, ибо речь идёт о методах человеческой деятельности, как научной, так и инженерной) работа ведётся в рамках комитетов по стандартизации, которые коллективно обсуждают различные догадки и находки по формальному представлению (получению теории) систем.

В последнее время предприняты некоторые попытки разобраться с понятиями системноинженерного проекта (в рамках работ Русского отделения INCOSE и SEMAT). Настоящий курс использует все эти наработки (http://www.slideshare.net/ailev/course-essence-sysengomgsep13), задавая научное основание для описания практик системной инженерии.

Само развитие системной инженерии, конечно, проходит от совершенно неформального эвристичного знания ко всё более и более формальному знанию. Можно условно поделить системную инженерию на четыре поколения, каждое из которых можно представить в виде какой-то отдельной линии развития — так называемой S-образной кривой. По вертикали указываются достижения, а по горизонтали — время. Сначала каждая из инженерных технологий развивается медленно, ищутся эффективные методы работы, а потом происходит бурный рост — эффективность технологии (понимаемая по-разному, в случае системной инженерии проще всего считать достижимую сложность целевой системы) уменьшается, и нужно переходить на следующее поколение технологии.




Первое поколение — это алхинженерия (”алхимическая” инженерия), по аналогии с алхимией по сравнению с химией. Помните алхимические неформальные описания химических реакций в те времена, когда не было ещё развитой химической нотации и понимания различий между химическими элементами и сложными веществами? Вот пример (http://gothicsstyle.ru/2011/02/02/srednevekovaya-alximiya-recepty/): *«Чтобы приготовить эликсир мудрецов, или философский камень, возьми, сын мой, философской ртути и накаливай, пока она не превратится в зелёного льва. После этого прокаливай сильнее, и она превратится в красного льва. Дигерируй (*нагревание твёрдого тела с жидкостью, не доводя её до кипения*) этого красного льва на песчаной бане с кислым виноградным спиртом, выпари жидкость, и ртуть превратится в камедеобразное вещество, которое можно резать ножом. Положи его в обмазанную глиной реторту и не спеша дистиллируй. Собери отдельно жидкости различной природы, которые появятся при этом. Ты получишь безвкусную флегму, спирт и красные капли. Киммерийские тени покроют реторту своим темным покрывалом, и ты найдёшь внутри неё истинного дракона, потому что он пожирает свой хвост. Возьми этого чёрного дракона, разотри на камне и прикоснись к нему раскалённым углём. Он загорится и, приняв вскоре великолепный лимонный цвет, вновь воспроизведёт зелёного льва. Сделай так, чтобы он пожрал свой хвост, и снова дистиллируй продукт. Наконец, мой сын, тщательно ректифицируй, и ты увидишь появление горючей воды и человеческой крови (Dumas, 1837, с. 30).*

Современное пояснение: “*Философская ртуть — свинец. Прокалив его, получаем массикот (желтую окись свинца). Это зеленый лев, который при дальнейшем прокаливании превращается в красного льва—красный сурик. Затем алхимик нагревает сурик с кислым виноградным спиртом — винным уксусом, который растворяет окись свинца. После выпаривания остается свинцовый сахар — нечистый ацетат свинца (чистый $Pb(C_2H_3OO)_2$ $3H_20$—это бесцветные прозрачные кристаллы). При его постепенном нагревании в растворе сперва перегоняется*

*кристаллизационная вода (флегма), затем горючая вода—«пригорелоуксусный спирт» (ацетон) и, наконец, красно-бурая маслянистая жидкость. В реторте остаётся чёрная масса, или чёрный дракон. Это мелко раздробленный свинец. При соприкосновении с раскалённым углём он начинает тлеть и превращается в жёлтую окись свинца: чёрный дракон пожрал свой хвост и обратился в зелёного льва. Его опять переводят в свинцовый сахар и повторяют все вновь.*

Это общий стиль мышления, речь тут не идёт только об алхимии. С любым знанием сначала так: хорошо ещё, что эти неформальные рассуждения вообще можно записать. Искусство создания более сложных объектов (например, каравеллы) было ровно что искусством: передавалась какая-то традиция, чертежей как таковых не было — обходились эскизами и макетами, пространными текстовыми описаниями, передававшимися зачастую даже не в книгах, а в рассказах — от мастера-инженера ученикам.

Классическая системная инженерия использует диаграммную технику — это уже не вольные поэтические метафоры, как в алхинженерии, но много более строгие определения системы: чертежи, диаграммы, таблицы и т.д. Но это не полностью формальное описание: его нельзя как-то формально проверить, оно предназначено для чтения и интерпретации только людьми. Если уподобить описание системы компьютерной программе по изготовлению системы, то это такая "программа", которую может выполнить только человек, но не станок-компьютер. Можно назвать это "псевдокодом": непосвящённый человек легко спутает псеводокод с компьютерной программой, но программист понимает, что псевдокод пишется для других людей, а не для компьютера. От псевдокода до реальной программы, реального формального текста на каком-то языке программирования примерно столько же работы, как от общего неформального понимания ситуации человеком до написания программы на псевдокоде.

Поэтому появляется третье поколение системной инженерии, моделеориентированная системная инженерия. Она предусматривает использование логических (структурных) и физических (числовых) формальных моделей, которые могут непосредственно быть обработаны (проверены, оптимизированы) компьютером. Это позволяет достигать принципиально другой сложности целевых систем: компьютеры проверяют модели на отсутствие разного рода ошибок в разы более производительно и точно, чем это может сделать человек. Основной особенностью моделеориентированной системной инженерии является то, что используются не только численные физические модели, но и "логические" модели, использующие аппарат дискретной математики, плюс алгоритмические модели на языках программирования.

Четвёртое поколение связано с тем, что моделируется уже не только целевая система, но и сами системные инженеры — их творческие практики. Проводятся гибридные (статистические и логические одновременно) вычисления, характерные для программ искусственного интеллекта, а не нынешних программ физического и логического моделирования, при этом эти все вычисления-моделирования-оптимизации увязаны друг с другом. Особенность четвёртого поколения в том, что не только люди создают модели, а компьютер только проводит вычисления по этим созданным моделям, но и компьютерные программы создают модели: компьютер выполняет творческие функции, которые сегодня выполняет системный инженер. Человек работает в партнёрстве с компьютером, а не программирует компьютер.

### Системный подход как научное основание системной инженерии

Если договориться о том, что системная инженерия должна быть дисциплиной не хуже, чем другие общепризнанные науки (например, физика и химия), то системной инженерии нужно найти свои теоретические понятия: основной объект изучения, который и задаёт всё многообразие дисциплины, всё многообразие изучаемого предмета. Тем, чем для механики является физическое тело, а для химии является химическая связь, в системной инженерии является система: научным основанием системной инженерии является системный подход.

В системном подходе главным понятием является "система". Точно так же, как все предметы реальной жизни (компьютеры, здания, пули, пыль) в механике отождествляются к физическим телам, а взаимодействие веществ в химии сводится к образованию и разрушению "химических связей", в системной инженерии рассматривают весь мир состоящим из систем. И точно так же, как в механике после отождествления пули или компьютера с физическим телом можно применить формулы для расчёта траектории падения компьютера или пули со стола (ибо это отождествление оставляет только важные для расчёта характеристики — прежде всего массу, но игнорируя цвет, собственников, основное назначение, причину падения и т.д.), так и в системной инженерии после называния чего бы то ни было "системой" можно применять известные способы рассуждения, известные "формулы" (хотя напомню, часто это не формулы надёжной теории, а просто хорошо зарекомендовавшие себя эвристики).

### Системноинженерное мышление коллективно

Ещё одной особенностью теоретической основы системной инженерии является то, что она должна учитывать коллективный характер человеческой деятельности. То есть понятие "система" каким-то образом должно быть увязано с другими понятиями, имеющимися в инженерном проекте — это явно не понятие "система" в безлюдном мире типа мира естественных наук. Нет, системноинженерное мышление должно учитывать существование людей, оно должно облегчать согласование многочисленных людских интересов по поводу создания успешных систем, должно облегчать коллективную работу.

Это означает, что в основе системноинженерного мышления должно быть целостное представление о человеческой деятельности (т.е. повторяющихся, типовых, присутствующих в культуре способах достижения цели — отдельное уникальное "действие" ведь "деятельностью" не назовут) по созданию успешных систем.

Системноинженерное мышление должно помогать размышлять не только о собственно целевой системе инженерного проекта (подводной лодке, компьютере, атомной электростанции, медицинском приборе), но и о системе деятельности ("проекте", обеспечивающей системе), которая создаёт эту целевую систему.

Тем самым в основании системноинженерного мышления лежат:

- Системный подход (как думать о системах)
- Ситуационная инженерия методов (как думать о деятельности)

### А в чём мышление?

Мышление появляется там и тогда, где и когда нужно решить проблемы — что-то, что непонятно как решать. До этого момента можно не мыслить, можно

заимствовать какие-то справочные решения, использовать уже имеющиеся знания, "на автомате". Хорошо сформулированная проблема обычно в себе содержит явное противоречие, которое необходимо "снять" — только в этот момент включается мышление, только в этот момент нужно "сесть и подумать" (а не "вспомнить и применить"). Иногда говорят, что мышление появляется, когда нужно "перевести проблемы в задачи", т.е. создать список работ, которые понятно, как делать, и которые вместе решают проблему.

Системная инженерия тут неоригинальна, решение проблем путём формулирования и снятия противоречий (коллизий) присуще и теории ограничений Элияху Голдратта ("грозовая туча"), и ТРИЗ+, и системомыследеятельной методологии (школа Г.П.Щедровицкого).

Системная инженерия ничего не говорит про то (не предлагает никаких "методов творческого мышления", таблиц решений, способов развития воображения), как снимать противоречия. Чудес не бывает, думать тут приходится не меньше и не больше, чем в любых других школах мысли. Системная инженерия позволяет удерживать вИдение всей системы в целом при решении проблем, не терять за деревьями леса, не терять за листьями дерева.

Системноинженерное мышление позволяет систематически (ежедневно, ежечасно) находить такие противоречия, требовать их решения, документировать эти решения. Системная инженерия, конечно, содержит знания по "типовым инженерным решениям", поощряет задействование опыта этих инженерных решений. Но когда вам нужно что-то делать впервые в мире (как когда-то летели на Луну, а сейчас делают первые возвращаемые на Землю повторноиспользуемые системы), то есть два варианта — изобретать что-то беспорядочно, или мыслить системно (системноинженерно), чтобы как-то последовательно ставить и решать проблемы, находить и решать противоречия.

Системноинженерное мышление как минимум помогает поделить решение проблемы между разными людьми в инженерном коллективе (более того, часто решение не может быть найдено одним гениальным человеком, требуется работа больших коллективов). Для этого системные инженеры явно обсуждают метод своей работы. Так, они не просто "генерируют основные инженерные решения", а "создают архитектуру системы" — профессиональный язык системных инженеров (он основан на системном подходе) позволяет быстрее договариваться о том, что делать и о чём думать, чем при использовании бытового языка.

Итого: системноинженерное мышление ничего не говорит про содержание мышления, только про его форму. Более того, системная инженерия делает всё, чтобы не нужно было мыслить, а нужно было бы просто применять в проекте уже известные технические решения. Но мощь системной инженерии будет проявляться в тот момент, когда известных технических решений не будет и нужно делать первую из данного вида (first of a kind, FOAK) систему, или обходить какие-то жёсткие ограничения, которые не встречались раньше.

### Наука/менеджмент = наука/инженерия

Менеджмент связан с системами особого рода — организациями, которые состоят не только из зданий, оборудования (капитала), не только из запасов материалов и денег, но и из людей. Часть менеджерской профессии при этом — инженерия предприятий: замысливание (conception), проектирование, эксплуатация, модернизация и ликвидация предприятий. Конечно, работа с людьми

принципиально отличается от работы с железом или программным обеспечением. Но параллелей между инженерным и менеджерским мышлением много больше. Настолько больше, что правильно было бы говорить об общем для них системном мышлении, а не системноинженерном мышлении. И знание о том, как описывать деятельности для них тоже будет общим.

По большому счёту, к менеджменту более-менее приложимы те же рассуждения, которые приведены тут про инженерию. Менеджмент тоже ориентирован на синтез, а не на анализ. Он основан на эвристиках, и использует теории тогда, когда эти теории есть. Конечно, значительная часть этих эвристик отличается от инженерных эвристик. Более того, менеджерские эвристики не записывают в толстые справочники (хотя посмотрите на многочисленные регламенты по организации работ в любом предприятии — не напоминает инженерные справочники? Единственное отличие, что такие справочники готовят на каждом отдельном предприятии, редко используются общие справочники для многих предприятий).

Так что материал нашей книги в части того, как описывать мышление и деятельность, во многом применим и к менеджменту, хотя и с некоторыми оговорками.

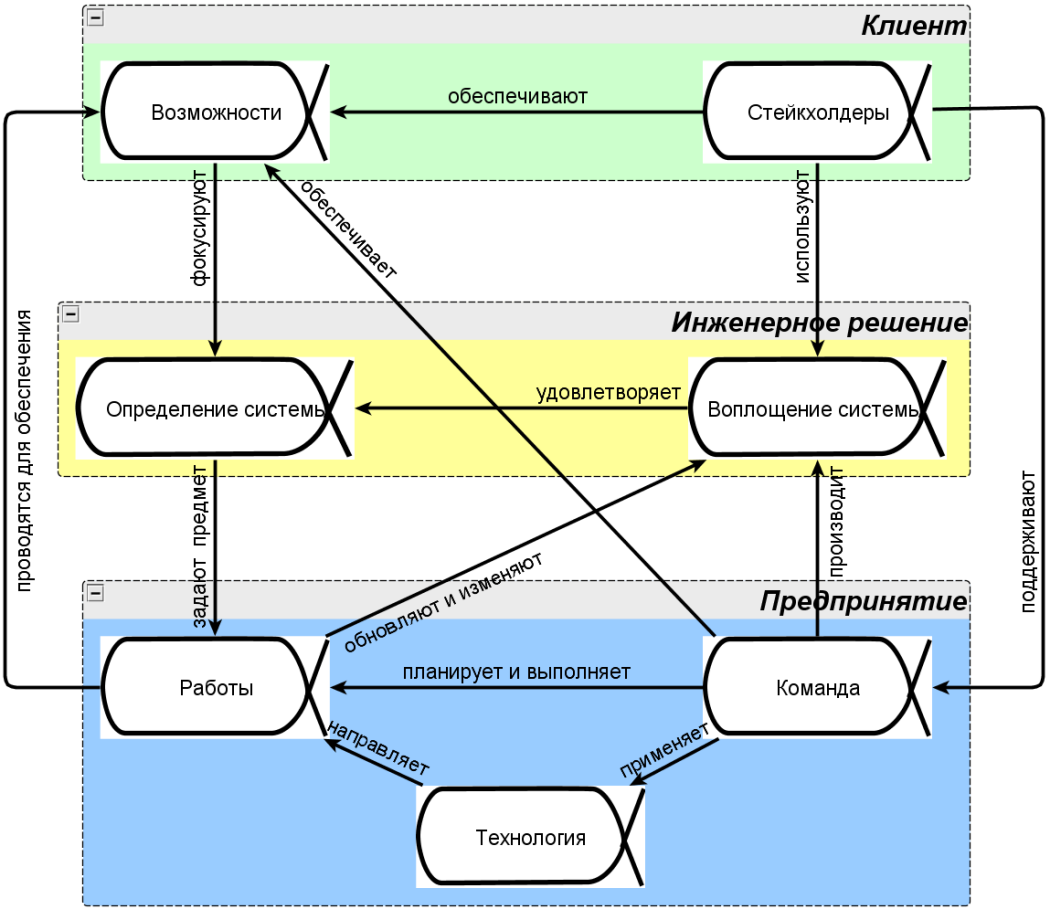
## 4. Схема/онтология инженерного проекта

В нашей книге по системноинженерному мышлению нас будет интересовать прежде всего ответ на вопрос "что есть в инженерном проекте?" (онтология) и "как формально и компактно это записать?" (онтологическое описание). В практиках моделеориентированной системной инженерии нас будет интересовать, что мы делаем с этими объектами. Но перед ответом на вопрос, что мы с этими объектами делаем, нам вначале нужно определить — какие там объекты, научиться находить эти объекты в жизни.

Философская линия Гуссерля-Витгенштейна-Бунге гласит, что знание выражается фактами: мы ничего не можем сказать про объекты мира, обозначая/представляя их понятиями языка, но мы можем представлять отношения объектов мира, представляя их выраженными в понятиях фактами. Следуя этой линии рассуждения, мы не можем давать определения отдельным объектам сами по себе, мы можем определять их только как входящие в какую-то схему из объектов и отношений. Эта схема мышления задаёт набор фактов.

Так, для определения понятия "целевая система инженерного проекта" нужны другие понятия ("стейкхолдер", "воплощение системы", "работы", "возможности" и т.д.) и ряд фактов о них в формате "объект1 в отношении X с объектом 2" ("стейкхолдеры используют воплощение системы", стейкхолдеры используют возможности, "работы проводятся для реализации возможностей").

Например, схема инженерного проекта предполагает набор из связанных отношениями семи понятий (concepts), каждое из которых обозначает что-то, что нас по поводу инженерного проекта интересует в реальности, изменения чего мы отслеживаем:




Это главная схема нашей книги, запомните её. Название тут неважно: схема инженерного проекта, она же диаграмма семи альф (обратите внимание, в узлах графа специальные значки «альфа»), она же диаграмма Основ системной инженерии (systems engineering Essence, от OMG Essence — “основа”, имени стандарта, где подобная диаграмма была предложена, или от kernel – «основа», «ядро» основных инженерных сущностей, предложенное стандартом), она же диаграмма инженерной деятельности, она же онтология инженерного проекта, она же одна из схем системного мышления.

На этой диаграмме основ отражены основные объекты, за изменением которых следят менеджеры и системные инженеры, и которые всегда присутствуют в их мышлении. Это не “реальные предметы”, это абстрактные сущности (типа “физическое тело”, “химическая связь”), но с этими сущностями как раз и проводятся реальные размышления — точно так же, как механик, вычисляющий траекторию выпущенной из ружья пули или летящей от пинка поручика Ржевского болонки абстрагируется от сущности летящих предметов и размышления свои ведёт в терминах “физического тела”, про которое ему известны формулы. Так и в инженерном проекте: системный инженер размышляет в терминах определения и воплощения системы, а не в терминах конкретных целевых систем (которых у него за долгую инженерную жизнь перед глазами пройдёт множество — как пациентов перед врачом. Да, каждый пациент конкретен, но учат врача работать с пациентами как абстрактными объектами, а не конкретными людьми — конкретные люди меняются, но знания о них, как о пациентах, у врача более-менее стабильны).

Что мы обсуждаем по схеме инженерного проекта:

- О чём в проекте нельзя забывать

- Где границы инженерного проекта, отделяющие его от других проектов
- Кто в проекте за что ответственен
- Какие максимальные риски, которые на себя может взять команда и её отдельные члены
- В каком состоянии сейчас проект, что уже сделано и что нужно ещё сделать для получения успешной системы
- … многое другое, ибо эта диаграмма отражает основные изменяющиеся в ходе проекта сущности и основные связи этих сущностей.

Рекомендуется эту диаграмму распечатать как плакат и повесить на стенку в том помещении, где совместно работают или совещаются менеджеры и системные инженеры. Это должно гарантировать, что при размышлениях о "воплощении системы" не будут забыты "стейкхолдеры", при обсуждении "команды" не будут забыты "технологии" и т.д.: схема задаёт некоторую мыслительную конструкцию, которой необходимо следовать в рассуждениях. Это не теория, использование данной схемы должно быть практикой.

Даже если мы ничего не можем сказать о «воплощении системы», мы из схемы уже знаем много фактов: "стейкхолдеры используют воплощение системы", "команда производит воплощение системы", "воплощение системы удовлетворяет определению системы" и т.д.

После знакомства с этой схемой затем в любом рассказе о "воплощении системы" мы, следуя этой схеме, будем искать в жизни "команду", "стейкхолдеров", "определение системы" и т.д. — под какими бы конкретными словами (терминами) в жизни они ни скрывались и какими бы вещами, явлениями, процессами или свойствами ни оказывались. Онтология — это то, что есть в жизни. Более того: схемы не только (и даже не столько) отражают то, что есть в жизни "на самом деле" — они предписывают то, что мы будем находить в жизни, ибо современная философия настойчиво намекает, что никакого "на самом деле" в жизни нет, а есть просто разные онтологии, описывающиеся разными онтологическими диаграммами/схемами.

Это, конечно, не значит, что на терминологию вообще не нужно обращать внимания. Так, в схеме инженерного проекта "предприНятие" — это не ошибка, и использование такого термина должно насторожить внимательного читателя. Действительно, этот термин обозначает другое, нежели традиционно понимаемое «предприятие-юрлицо», но об этом позже.

### *Схемное/онтологичное мышление*

Конечно, всей сложности жизни одной схемой/онтологической диаграммой не отразишь, схем приходится иметь множество — и трудность мышления даже не в использовании схемы, а в том, что схем огромное количество и они тесно перевязаны друг с другом. Рельс мышления огромное количество, и между ними множество связей — и всё одно мысли передвигаться по этим рельсам быстрее, чем каждый раз изобретая устройство мира (онтологию) сообразно ситуации. Вряд ли вы по-быстрому изобретёте системную инженерию, системноинженерное мышление, системный подход. Так что лучше их не изобретать, а приучить себя думать согласно схемам. Так, схема инженерного проекта (она же онтология инженерной деятельности) взята из немного доработанного стандарта OMG Essence. Вряд ли вы придумаете такую схему быстро, легче её позаимствовать из

стандарта/книги (такие "заимствования" разработанных в других инженерных или научных проектах знаний в инженерии называют библиотечными/library в противопоставление тем проектным знаниям, которые вырабатываются прямо по ходу инженерной работы в самом проекте, и которые потом можно будет использовать в других проектах).

Согласно схеме, в инженерном проекте есть "команда" и "возможности", "работы" и "технология". Поймёте ли вы, что это высказывание относится к предъявленной схеме инженерного проекта, если термины-обозначения не выделять кавычками? Сравните: "Итак, в инженерном проекте есть команда и возможности, работы и технология" — понятно ли, что фраза составлена с жёсткой опорой на схему инженерного проекта? Контрольный вопрос: что ещё есть в инженерном проекте? Если вы помните о схеме инженерного проекта, то ответы "журнал учёта посещаемости", "испытательные стенды" и "три инновации" не пойдут в зачёт, даже если они там есть и они важны. Как на вопрос "что у нас сегодня на обед" отвечают про первое, второе и третье блюдо в целом, но не "вишня в компоте" (хотя эта вишня, безусловно, входит в обед), так и в инженерном проекте отвечают прямо по схеме инженерного проекта, прямо по диаграмме, прямо в терминах задаваемой диаграммой онтологии (того, что есть в мире — по мнению диаграммы и согласных с ней людей): ещё в инженерном проекте есть стейкхолдеры, определение системы и воплощение системы. Поменяете схему — будут другие ответы, но пока мы работаем с этой схемой — в зачёт идёт только этот ответ. Нет никакого "на самом деле", есть только "в соответствии с выбранной нами онтологией/схемой".

Схемное мышление и использование схемы может быть совершенно незаметно постороннему взгляду, но знающим схему людям оно хорошо заметно! И помним про терминологию: слова не так важны, важны понятия, которые обозначаются словами — и эти понятия как раз и задаются схемой, не давайте себя поймать в ловушку слов. Так, вместо "команды" может быть "рабочая группа проекта" или "интегральный коллектив разработки" — если вы мыслите по схеме, то вы легко поймёте, о чём речь — и все ваши знания про "команду" будут готовы к использованию в том проекте, в который вы попали, и терминологию которого вы не контролируете.

В книге мы будем использовать онтологические описания/схемы/диаграммы, отражающие основные онтологические факты об отношениях объектов в предметной области системного мышления, системной инженерии, ситуационной инженерии методов и других нужных нам дисциплин. Именно эти схемы/понятийные диаграммы/онтологические описания дают ответ на вопрос "что есть в мире, что относится к системной инженерии?". Рассуждения в книге будут строиться на основе таких схем/понятийных диаграмм/онтологических описаний. После некоторого упражнения такие схемы формируют в мозгах "рельсы мышления", которые позволяют думать быстро — не изобретать велосипед каждый раз, когда вам нужно подумать, например, об инженерном проекте. Если вы подумали про "воплощение системы", то у вас немедленно в сознании должны всплыть схемы, где есть "воплощение системы", и из этих схем (например, схемы инженерного проекта) выпрыгнуть в зону повышенного внимания также и "стейкхолдеры" — ибо из схемы пришло отношение "стейкхолдеры используют воплощение системы". Это всё должно быть натренировано буквально до бессознательного уровня, как вы когда-то в первом классе натренировали 2*2=4 и "Волга впадает в Каспийское море".

В книге мы будем мягко относиться к формально-логической стороне дела и не

будем требовать полностью формальных (понятных даже компьютерам) суждений, но следование в рассуждениях предлагаемым схемам мышления необходимо. Системноинженерное мышление и деятельность в своей основе направляются привычными заученными схемами. Схемность мышления вовсе не означает его ограниченности. Просто объекты реального мира "подводятся под понятия" никогда не по одному, а целыми группами, связанными отношениями — то есть под понятия подводятся сразу ситуации, между объектами реального мира предполагаются прописанные в схеме отношения. Если схема кажется не подходящей для ситуации, её можно как-то адаптировать — создать новый набор фактов, более адекватно описывающий мир, более адекватно отражающий устройство мира. Но перед тем как это делать, научитесь сначала пользоваться уже наработанными схемами, не изобретайте велосипедов, не будьте Кулибиным-самоучкой. Вы вряд ли гениальный онтолог, поэтому смело вставайте сначала на плечи гигантов. И когда вы научитесь мыслить и действовать в рамках уже существующих схем, вы ясней поймёте их достоинства и недостатки, сможете эти недостатки преодолеть — предложите способы системноинженерного мышления лучше, чем имеющиеся сейчас, т.е. выпустите свой международный стандарт с вашей новой схемой, новым вИдением мира, новой онтологией.

Те, кто знают схему — это и есть "семантическое сообщество". Они знают, что "команда применяет технологии", как бы ни называлась "команда" (бригада, компания, рабочая группа", как бы ни произносилось "применяет" (использует, опирается на, задействует, владеет), как бы по-разному ни появлялись бы в речи "технологии" (способы работы и инструменты, методы и их поддержка, производственная среда).

Итого: системноинженерное мышление — это умение рассуждать с использованием базирующихся на системном подходе и относящихся к инженерным проектам онтологических схем. Это умение подводить объекты реального мира под понятия на схемах и вести рассуждения с использованием этих понятий независимо от используемой в том или ином речевом сообществе терминологии.

### *Ситуационная инженерия методов*

Для того, чтобы обсуждать, как устроено мышление системного инженера, нам нужно схемно/онтологически задать "теорию", способ компактного описания того, что происходит в инженерных проектах: ввести основные понятия, которые должны присутствовать в нашем мышлении в каждом инженерном проекте. После этого мы получим способ обсуждать происходящее в разных проектах, используя эти основные понятия.

Описанием хода инженерной разработки (development process, engineering methodology) занимаются в рамках дисциплины "ситуационная инженерия методов" (situational method engineering). Она была основана в начале 80-х годов идеологами объект-ориентированного движения в программной инженерии, которые задали два основных структурированных (ибо неструктурированные в форме "просто текста на естественном языке" никто не отменял) вида описания своих способов работы:

- использование "языков паттернов" (ищутся некоторые "паттерны" — неформально определяемые способы решения задач, при этом каждый паттерн описывается по заранее известному шаблону, в который обычно входит описание проблемы и типовой способ её решения). Ассорти ссылок про языки паттернов тут: http://ailev.livejournal.com/487783.html. Паттерны — это чистой воды эвристики, никаких попыток выйти на какие-то более-менее

формальные “языки паттернов” не делалось. Само слово “язык” в словосочетании “язык паттернов” используется неформально (просто чтобы указать на то, что в проекте используются разные паттерны в разных сочетаниях, как слова из какого-то языка).

- ситуационную инженерию методов, как дисциплину. Стандарты описания метода в такой дисциплине обычно представляет собой “мета-модель”: описание языка, используемого для моделирования способов работы.

Donald Firesmith рассказывал, что он с друзьями занимались в начале 80-х объект-ориентированным программированием, что тогда было остромодно и ново. Как-то его вызвал начальник и сказал: “у тебя производительность программирования стала в последнее время в несколько раз выше, чем у остальных в нашей компании. Научи их своему методу работы”. Дональд пошёл на своё рабочее место думать над новым заданием, и понял, что не понимает — что такое “метод работы”, чему же он должен научить других сотрудников? Что происходит в ходе выполнения программистского проекта методом объектно-ориентированного программирования? Ему было понятно, что он работает уже не так, как до знакомства с объектно-ориентированным программированием, но как описать эту разницу в работе? Внешне ведь это выглядело просто как “сижу и думаю”, описывать нужно было не столько внешнее поведение, сколько происходящее “в мозгах” — да ещё и не само поведение, а “метод работы”, задающий поведение для многих и многих разных проектов.

Так он с друзьями начал разрабатывать новую дисциплину “инженерия методов”. Скоро выяснилось, что никакой метод работы не может быть перенесён из того конкретного проекта, в котором он был разработан, в другой проект. Ситуация (технология — инструменты и рабочие продукты, используемые в конкретном предприятии, и особенности целевой системы в конкретном проекте) меняла всё заранее записанное поведение-образец. И тогда инженерию методов переименовали в ситуационную инженерию методов (situational method engineering) — чтобы подчеркнуть тот факт, что каждый метод работы зависит от ситуации, а между ситуациями выживает не сам метод как предзаданная последовательность операций, а только какие-то его части. В разных школах ситуационной инженерии методов эти части назывались компонентами/components, кусочками/chunks, ломтиками/slices и т.д. — главным образом компонентами выступали “артефакты” (рабочие продукты — над чем работаем), “процессы”, “инструменты” и т.д.

Языки ситуационной инженерии методов, каждый из которых определял свой набор “компонент метода” оформлялись в виде стандартов, по которым далее разрабатывались описания самых разных методов. Этих методов инженерной работы (описаний того, как нужно начинать работу в инженерном проекте и как нужно заканчивать) существует огромное количество — их десятки тысяч. Появились два поколения ситуационной инженерии методов и отражающих эти поколения стандартов

- первое, ориентированное на методологов, которым нужно было описывать методы работы и систематизировать эти методы работы. Это прежде всего стандарты OPF, ISO 24744 и OMG SPEM 2.0. Обзор этого первого поколения дан в http://www.jucs.org/jucs_16_3/situational_method_engineering_state/jucs_16_03_0424_0478_henderson.pdf
- второе, ориентированное прежде всего на удобство пользователей описаний

(инженеров), а не на методологов. Пока это поколение стандартов состоит из стандарта OMG Essence (http://www.omg.org/spec/Essence/ — версия 1.0 была выпущена в ноябре 2014).

Описание метода в настоящем курсе системноинженерного мышления

Настоящий курс системноинженерного мышления будет использовать адаптированный (существенно упрощённый, изменённый для работы с системноинженерными, а не софтверными проектами, а также переведённый на русский язык) стандарт OMG Essence — http://www.omg.org/spec/Essence/. Этот стандарт разработан в рамках инициативы SEMAT (http://semat.org).


Утверждение его прошло в консорциуме по стандартизации OMG (Object Management Group, http://www.omg.org).


Адаптация для системной инженерии проводилась TechInvestLab (http://techinvestlab.ru).


Обсуждение этой адаптации и перевода на русский язык проходило на заседаниях Русского отделения INCOSE (http://incose_ru.livejournal.com), идёт международная дискуссия, для чего был выпущен на английском языке продукт Русского отдления INCOSE “Towards Systems Engineering Essence” — http://arxiv.org/abs/1502.00121


Стандарт предусматривает описание основных видов компонент метода (на языке компьютерного моделирования, текстовом языке и в графической нотации) и правила описания методов с использованием этих компонент. Стандарт определяет понятия “практики”, “метода”, “рабочего продукта”, “альфы” и т.д.

Кроме того, стандарт предлагает описание основных понятий программной инженерии как дисциплины (kernel, мы переводим это как “дисциплина”): те объекты, с которыми программисты встречаются практически в каждом инженерном проекте. Так, стандарт определяет такие “альфы Основы”, как “стейкхолдеры”,

“возможности”, “работы”, “команда” и т.д.

Данный стандарт в настоящем курсе:

- Используется не целиком. Мы сосредотачиваемся в нём главным образом на альфах и немножко на рабочих продуктах, остальное игнорируем.
    - Альфы инженерного решения мы заменяем с предложенных стандартом для программных проектов “требований” и “программной системы” на системноинженерные альфы “определение системы” и “воплощение системы”.
- Пересказан по-русски так, чтобы он был гармонизирован по терминологии с остальным содержанием курса (помним, что нас больше волнует “сообщество значений”, нежели “речевое сообщество” — мы за сохранение смысла и понимания, а не за сохранение слов-терминов. Но это не значит, что русскоязычная терминология нас вообще не волнует!).

Яблоки из жизни и яблоки из задачи

Вы будете удивляться, но это ключевой раздел всей книги: **в жизни нет ни одного слова из книги, в книге нет ни одного слова из жизни** — и в этом разделе поясняется, почему, и что с этим делать.

В основу описания инженерной деятельности мы положим такие теоретические объекты, как “альфы инженерного проекта”, тесно связанные между собой:

Главная трудность в понимании альф — это как основы системной инженерии (теория, содержимое нашей книги) связаны с практикой, с реальным миром. В реальном мире мы видим конкретные предметы, с которыми работает инженер: рабочие продукты (часто используемый синоним — “артефакты”, artifacts, т.е. предметы искусственного происхождения, work products). Эти рабочие продукты мы можем пощупать, пнуть ногой, указать на них пальцем. В реальном мире мы видим артефакты “яблоки” и дела с этими артефактами — “сосчитать яблоки”. Альфы представляют собой те объекты, которые описываются дисциплиной (теорией) — “количество предметов” и “сосчитать предметы”, если вернуться к примеру с яблоками. Тут нужно привести байку про яблоки из задачи и из жизни, например, в таком варианте:

> пришла в ... школу учительница, которая начиталась работ о дидактической функции наглядных пособий и считала, что надо учить на наглядных пособиях. А проходили они в этот момент задачку на сложение: «3+5». И она принесла три яблока и ещё пять яблок, выложила их на стол и говорит: «Дети, вот вы видите здесь — раз–два–три — три яблока, а здесь вот — раз–два–три–четыре–пять — пять яблок. Вот я их соединяю, сколько получится всего яблок?» Дети пялятся на яблоки, слюни у них текут, но задачи не понимают. Второй день проходит, третий — рекорд: в таком классе обычно за день это проходили. Она приходит в учительскую, жалуется, что вот–де она применяет новые методы, наглядно все, а результата нет. И вот на пятый день с задней парты тянется рука, и ученик говорит: «Мэм, я теперь понял: эти яблоки, которые вы выложили на стол, не настоящие — это яблоки из задачи». — «Да, а что?» — «Ну тогда, мэм, совсем другое дело». И с этого момента, когда класс понял, что это не настоящие яблоки, а яблоки из задачи, все моментально пошло. Почему? Когда вы кладёте реальные яблоки — что с ними надо делать? Их надо есть. А

чтобы считать, нужны рисуночки.

Вот ещё про то же самое:

– Мы займёмся арифметикой... У вас в кармане два яблока...

Буратино хитро подмигнул:

— Врёте, ни одного...

— Я говорю, — терпеливо повторила девочка, — предположим, что у вас в кармане два яблока. Некто взял у вас одно яблоко. Сколько у вас осталось яблок?

— Два.

— Подумайте хорошенько.

Буратино сморщился, — так здорово подумал.

— Два...

— Почему?

— Я же не отдам Некту яблоко, хоть он дерись!

— У вас нет никаких способностей к математике, — с огорчением сказала девочка.

Обычно занимающиеся по нашей книге проходят следующие стадии при изучении системноинженерного мышления:

- Ничего не понимают, ибо неспособны соотнести материал книги с реальными проектами. Действительно, в их проектах нет никаких альф! А в книге нет ничего, что есть в их проектах (более того, каждый проект уникальный — в них ведь вообще нет ничего общего)!
- Всё понимают про приводимые примеры, про проекты однокурсников и коллег по работе, ничего не понимают про свои собственные проекты. Конечно, ибо чужие проекты — это "проекты из задачи" (помним, "яблоки из задачи — их можно считать"), а свои проекты — это "реальные проекты" ("яблоки из жизни", их нужно есть!).
- Всё понимают и про свои проекты, и про проекты коллег. Но ничего из понимаемого не делают (ибо системноинженерное мышление изучается не для того, чтобы его применять, а "для самообразования и развития", "для сдачи зачёта" и т.д. Применять знания мешают начальники, текучка, лень, отсутствие единомышленников, помогать применять знания некому).
- Применяют материал книги в своих проектах, ибо так работать оказывается качественней, легче и в конечном итоге быстрее. Очень часто это происходит только через год-два после знакомства с материалом, не раньше.

### Альфы

Альфы — это яблоки из учебников и задачников, мыслительные операции делаются с ними. Рабочие продукты/артефакты — это яблоки, которые мы едим, их можно найти в проектах. К ним применяются разные действия. Как их различить, и как сопоставить друг с другом — это главная трудность в освоении не только системной инженерии, но и любой другой дисциплины, описывающей окружающий мир и его закономерности: как объекты любой теории совмещать с объектами реального

мира. Это как раз то, что физиков отличает от математиков: физиков волнует, чему в реальном мире соответствуют их формулы, а математиков не волнует.

На диаграмме альф инженерного проекта стейкхолдеры, возможности, определение и воплощение системы, команда, технология и работы — это альфы, а не рабочие продукты.

Альфы — это функциональные (выполняющие определённую функцию, играющие определённую роль, идеальные) объекты, по которым мы судим о продвижении (progress, "как много мы уже сделали?") и здоровье (health, "в проекте всё идёт хорошо") проекта. Альфы — это абстракция того же сорта, какого "физическое тело" является абстракцией реальных физических объектов (да, это физическое тело имеет массу, а геометрическая точка имеет координаты. Но мы связываем физические тела и математические точки как идеальные объекты с реальными объектами, и после надлежащего тренинга "склеиваем" в мышлении идеальные и реальные объекты. Поэтому об экземплярах альф в проекте принято говорить так, как будто они вполне реальны и существуют в мире, несмотря на все абстракции.

Альфы фиксируют компактное описание мира/теорию, удобную для решения каких-то практических проблем. Это нужно, чтобы иметь возможность повторно использовать известные нам способы рассуждений и решения задач для самых разных объектов. Так, мы думаем о "физическом теле" и "математических точках" единообразно, "как в учебниках физики и геометрии", а применимо это мышление к самым разным "реальным объектам вокруг нас" — от коробка спичек до галактик. В этой экономии мышления (учимся думать один раз, затем похоже думаем в самых разных ситуациях) и заключается смысл разделения альф и рабочих продуктов. Например, учимся думать о "требованиях" — а применяем потом это мышление к конкретным рабочим продуктам, которые можно найти на производстве "в реале": спецификациям требований, требованиям из текстов стандартов, user stories на карточках, записям в базе данных системы управления требованиями и т.д.

Несмотря на всю "идеальность" и "абстрактность", об альфах говорят как о вполне существующих в физическом мире — подразумевая при этом их тождественность тем рабочим продуктам, по которым мы можем судить об их существовании. Так, можно определить альфу "моя любимая игрушка" — и, хотя в детстве у меня это был нарисованный на ватмане пульт управления космическим кораблём, а сегодня это мой ноутбук, я могу говорить про прохождение "моей любимой игрушкой" состояний "полюбил", "разлюбил", "поломалась", "играю", "забросил" и т.д. — независимо от того, какая именно это игрушка прямо сейчас. Если я говорю о "требовании" — то меня не волнует, пункт ли это протокола совещания с представителями заказчика, или запись в базе данных системы управления требований, или фрагмент диаграммы какой-то модели требований. Для меня это "требование" — и я после этого знаю, что с ним делать, и как о нём думать, я обсуждаю "требование" как реальный объект, существующий в мире, имеющий своё состояние и меня в этот момент мало волнует, что у этого требования есть ещё и какие-то особенности выражения (как при счёте яблок меня мало волнует, что их ещё и едят).

Формально ALPHA это Abstract-Level Progress Health Attribute, но неформально это просто "идеальный рабочий продукт", названный "альфой" для уменьшения путаницы с "реальными рабочими продуктами" и аббревиатура для него была подобрана задним числом. Альфы — это то, что изменяется в проекте, и изменения чего мы хотим понимать, отслеживать, обеспечивать, направлять, контролировать.

То, что альфы (точнее, экземпляры альф) изменяются в ходе стратегирования, инженерной, организационной, операционной деятельностей, отражено в том, что альфы имеют состояния (state), а эти состояния имеют контрольные вопросы (checkpoint) для определения достижения этих состояний. Состояния альф обычно определяются на всём пути от самого появления альфы в проекте до прекращения её существования.

Пример: альфа "воплощение системы" имеет состояния "в виде сырья", "в виде частей", "демонстрируемо", "готово", "эксплуатируется", "выведено из эксплуатации". Контрольные вопросы для достижения состояния "готово" (система как целое была принята для эксплуатации):

- Функциональность, обеспечиваемая системой, протестирована.
- Уровни дефектов приемлемы для стейкхолдеров.
- Установочная и другая пользовательская документация доступна.
- Представители стейкхолдеров принимают систему, как удовлетворяющую своему назначению.
- Состав передаваемой стейкхолдерам системы известен.
- Представители стейкхолдеров хотят принять систему в эксплуатацию.
- Эксплуатационная поддержка наличествует.

Планирование и контроль выполнения инженерного проекта ведётся прежде всего в терминах этих состояний альф: каждый пункт плана и контроль выполнения плана имеет ввиду достижение этого состояния, а контрольные вопросы обеспечивают уверенность в отсутствии крупных промахов.

Альфы могут быть вложены друг в друга (связь AlphaContaiment, выражает отношение часть-целое двух альф), при этом выполняется важное правило: продвижение по состояниям вложенной альфы (части) означает какое-то продвижение по состояниям объемлющей альфы (целого). Так, продвижение требований как целого по их состояниям может быть за счёт того, что продвигаются отдельные требования — по их состояниям. Никакой связи между состояниями частей ("подчинённых" альф, subordinate) при этом и состоянием целого (альф-"начальников") нет, кроме общего понимания, что "продвижение в изготовлении частей как-то продвигает нас в изготовлении целого".

Отношения вложенности альф — это тоже направленный граф, одна альфа может быть частью других альф. Так, член команды может быть подальфой и команды в целом, и подальфой стейкхолдеров (ибо члены команды формально тоже стейкхолдеры, хотя и "внутренние"). Обратите внимание, что рассуждения про альфы тут такие же, как про реально существующие объекты. Яблоки в задаче и яблоки в жизни очень легко перепутать, для тренинга в их использовании и нужны учителя и консультанты.

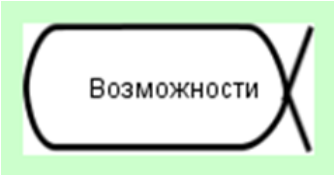
Альфы между собой связаны — можно думать об этих связях, как о стрелочках на диаграмме альф (например, "определение системы определяет воплощение системы", "команда выполняет работы" и т.д.).

Рабочий продукт/артефакт представляет (represent) собой альфу в реальном мире. Это "яблоко из жизни", "его едят" — рабочий продукт имеет в реальном мире какую-то пользу для проекта. Рабочие продукты — это спецификации, тесты, диаграммы и какие-то модели, базы данных и физические объекты. Рабочим продуктом могут

быть и какие-то мероприятия (совещания, презентации, уборка рабочего места), которые можно представлять “вещественно” как совокупность участвующих в этом мероприятии взаимно изменяющихся объектов. Одну альфу может представлять несколько рабочих продуктов, ибо у альфы в деятельности могут проявляться много различных свойств, требующих различных представлений в мире. И несколько альф могут быть представлены в одном рабочем продукте. Эта связь рабочих продуктов и их альф обычно определяется в рамках какой-то практики (выбранной технологии работы). Рабочие продукты ситуативны для каждой конкретной организации и даже конкретного проекта, а вот альфы более стабильны.

Экономия мышления заключается в том, что часто одна альфа представляет до десятка разных рабочих продуктов. Обсуждение и мышление тем самым ведётся только для одного объекта, и только при разбирательстве с какими-то конкретными деталями вытаскиваются отдельные рабочие продукты. Например, “воплощение системы готово к проведению пуско-наладочных работ?” — в то же время доказательство готовности воплощения системы к пуско-наладочным работам может быть разбросано (кроме факта наличия самих рабочих продуктов, представляющих воплощение системы “в металле”) по десяткам разных рабочих продуктов: документов типа актов сдачи работ различными подрядчиками, актов предварительных испытаний, писем контрагентов, записей в базах данных систем управления активами предприятия, сообщений о наличии расходных материалов и т.д.

Альфы обозначаются значком, напоминающим альфу (в диаграмме Основ системной инженерии именно эти значки), со вписанным названием альфы.




Рабочий продукт обозначается значком, напоминающим “документ” (лист бумаги с загнутым уголком), это должно напоминать о том, что он “присутствует в мире”:


Рабочие продукты могут быть как входными для проекта, так и представлять собой результаты, или же быть полезными только команде, не являясь ни входными, ни результирующими для проекта.

Как и альфа меняется, проходя состояния, рабочий продукт меняется, проходя уровни детальности. Рабочие продукты в ходе инженерного проекта создаются, модифицируются, уничтожаются. Они вместе с их уровнями детальности наблюдаемы, в отличие от альф, о состояниях которых мы можем судить только "по приборам" — по рабочим продуктам.

Уровни детальности также определяются контрольными вопросами, только в случае альф состояние определяется через "и" ("да" в ответе на все вопросы), а в случае

рабочих продуктов уровень детальности определяется по любому из вопросов, т.е. через "или" ("да" в ответе хотя бы на один вопрос). Состояния альф и уровни детальности рабочего продукта не связаны между собой, они про разное. Так, для согласованных со стейкхолдерами требований спецификация может быть в виде краткого резюме, более-менее подробна, в виде формальной модели. С другой стороны, могут быть детальнейшие требования, которые совсем не согласованы со стейкхолдерами, и эти требования могут быть противоречивы между собой. Так что уровни детальности — это просто уровни гранулярности и объёма информации, который мы можем почерпнуть о той или иной альфе, а уж какое состояние этой альфы — это другой вопрос. Заметим, что работа прежде всего продвигает состояние альф — а уж будет ли продвинуто состояние альфы, если вложиться в работы по повышению детальности рабочего продукта, это большой вопрос. Повышение детальности рабочих продуктов, конечно, как-то связано с общим состоянием работ в проекте, но не впрямую.

### Метонимия и схемы

Метонимия — это когда в речи один элемент схемы обзывается другим "по смежности" отношения в схеме (в то время как метафора — это про "похожесть" структур разных схем). Подробней: http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%BD%D0%B8%D0%BC%D0%B8%D1%8F

Очень часто метонимия возникает в части альф и рабочих продуктов: по самым разным видам отношений: то проект назовут командой ("погляди, в каком состоянии там третий отдел" — имеется ввиду не состояние команды, а состояние всех альф выполняемого третьим отделом проекта), то команду назовут системой ("что там у нас с отпусками по усилителям мощности?"), но чаще всего путают альфы и рабочие продукты. "Требования утвердили?" — в вопросе альфа "требования", хотя речь идёт о рабочем продукте-документе "опросный лист", часто использующемся в тендерах на проектирование и монтаж "под ключ" установок в топливо-энергетических проектах, а альфа "требования" отражается добрым десятком и других документов тоже.

Без метонимии нельзя разговаривать, речь ведь должна быть живой. Но нужно быть крайне внимательным, когда живую речь мы переводим в формальные диаграммы. Читаем схемы мы обычно с использованием метонимии. Писать схемы нужно, устраняя метонимию (это специальная работа онтологического моделирования). Понимать речь нужно, зная о метонимии и не попадаясь в её ловушку перескока с одного элемента схемы на другой по отношениям "род-вид" (специализации), "часть-целое" (композиции), "функционального объекта-конструктивного объекта" (установки, installed_as).

## *Методологическая действительность: дисциплины, практики, методы*

### Дисциплины/области интереса

Основная схема включает указание на три основные предметные области/области интереса/дисциплины (area of concern), они выделены на диаграмме зелёным, жёлтым и голубым цветами:

- Технологический менеджмент, упор на маркетинг/стратегирование: как сотрудничать со стейкхолдерами, которые дают возможности для самых разных проектов ("предпринятий" — от "предпринять что-то") и соединять их

с возможностями, которые даёт команда. Если возможности стейкхолдеров и команды совпали, то проекту быть! На диаграмме эта предметная область/дисциплина отмечено, как относящееся к "клиенту" (client). Системная инженерия всегда кого-то обслуживает! Инженеры не разрабатывают системы сами для себя!"

- инженерию: как придумать/определить и сделать/воплотить успешную систему. На диаграмме это область "инженерного решения" (solution, не путайте с decision!).
- Инженерный менеджмент, упор на операционный менеджмент и лидерство: как собрать и сплотить компетентную команду, оснастить её всеми необходимыми технологиями и организовать работы по проекту. На диаграмме это область "предпринятия" (от "предпринять" что-то, но слово "предприятие" не подходит, речь тут идёт не о юридическом лице — это может быть и большой холдинг, и какая-нибудь "рабочая группа проекта". В OMG Essence это "Endeavor", ибо стандарт писали в США главным образом. В Англии было бы "более по-французски" — Endeavour).

Каждая дисциплина включает в себя основные альфы этой дисциплины (так, инженерия работает над целевой системой, которая рассматривается как совокупность определения системы и описания системы. Эккаунт-менеджер (маркетолог, стратегический менеджер, менеджер по работе с клиентами) занимается стейкхолдерами и теми возможностями, которые эти стейкхолдеры приносят проекту (главным образом возможности приносят нужды стейкхолдеров-клиентов). Если речь идёт не о заказном проекте в уже имеющейся компании, то этим занимается технологический менеджер-предприниматель, и он же будет подбирать команду. Операционный менеджмент озабочен планированием и контролем выполнения работ, организацией команды и наличием технологии (чтобы команда работала не голыми руками — технология это инструменты, рабочие продукты и т.д.).

Каждая из основных дисциплин состоит из своих поддисциплин. Например, инженерия включает в себя поддисциплины:

- инженерия требований (требования — это часть определения системы)
- инженерию системной архитектуры (системная архитектура — это часть определения системы)
- системный дизайн (3D-модель — это часть определения системы)
- проверки и приёмка (соответствия определения и воплощения системы, воплощения системы и возможностей стейкхолдеров)
- управление конфигурацией (поддержка целостности определения системы и его соответствия воплощению системы)
- специальные инженерии (электрика, теплотехника, программная инженерия, ...), работающие со своими особыми частями определения и воплощения системы.
- ... (этот список далеко не полный).

Это, конечно, очень упрощённое представление о предпринятиии. Так, есть особенности для предпринятия как нового инвест-проекта и предпринятия для проекта разработки в уже давно состоявшемся предприятии. Технологиями обычно занимаются CTO (chief technical officer) и CIO (chief information officer), их часто

относят к технологическому менеджменту, а не инженерному менеджменту. Но тут важно понять идею: выделять отдельные области интересов и обсуждать их по отдельности (мыслительно «разделять и властвовать», принцип separation of concerns), не валить всё в кучу, предполагать какое-то разделение труда, наличие разных дисциплин.

### Практики

Каждая поддисциплина может разбиваться на поддисциплины дальше, но может становиться практикой, если добавляются знания о том, какие в предпринятии используются технологии (инструменты и рабочие продукты).

Тем самым в простейшем варианте (пока мы не обращаем внимания на дела, процедуры, операции и т.д.):

- Дисциплина = набор альф, уровень учебника (”дисциплина грузится в голову”).
- Технология = рабочие продукты (нотации моделей, формы документов, софт, станки, инструменты), по которым можно определять состояния альф. Технология разворачивается на предприятии, для этого менеджмент должен выделить необходимые ресурсы.
- Практика = дисциплины + технология.

Практика (practice) — это описание того, как справиться с каким-то отдельным аспектом (но не всеми!) дисциплин инженерного проекта. Практика может состоять и из других практик, но она всегда базируется на каких-то дисциплинах или поддисциплинах, то есть технология (рабочие продукты) может только дополнять альфы, а не подменять собой альфы.

Практика для какой-то цели помогает характеризовать проблему какой-то (маркетинговой, инженерной, менеджерской) дисциплины, даёт стратегию для решения проблемы, инструктирует как проверить, что проблема и в самом деле была решена. Если нужно, то практика также описывает, какие инженерные обоснования нужны, и как заставить стратегию работать в реальной жизни. Практика обеспечивает систематический и повторяемый способ работы (дисциплину мысли и технологию, развёрнутую на предприятии), фокусирующийся на достижении цели практики.

У практики есть свои правила обеспечения целостности (что необходимо и достаточно иметь и уметь, чтобы можно было достичь цель). Цель указывается лаконичной обособленной фразой (предложением), но могут быть даны и дополнительные объяснения.

Практика также имеет указание на единицы измерения, оценивающие результаты (performance) практики и меру достижения её цели. Измерения проводятся и документируются в ходе выполнения практики.

Практика имеет ещё и ожидаемые характеристики входных (имеющихся в начале выполнения работ практики) и выходных (с результатом после завершения выполнения работ практики) элементов. Критерий завершения практики формулируется в терминах состояний альф и уровней детальности рабочих продуктов этой практики.

Практики объединяются (composition) в другие практики, чтобы охватить больше аспектов. Когда результат такого объединения практик охватывает все аспекты

проекта, тогда это уже метод.

По-русски чаще всего описание практики оформляют в виде "методических рекомендаций", "методики".

Метод

Метод — это сочетание дисциплины и набора практик для достижения какой-то цели. В инженерии бывают методы разработки, методы сопровождения, методы интеграции систем, методы проведения испытаний, в операционном менеджменте бывают методы управления проектами, методы обеспечения цепочки поставок и т.д.

Цель метода должна быть сформулирована отдельно как короткое предложение, к ней возможны дополнительные описания. Эта цель должна учитывать нужды стейкхолдеров, условия ситуации и желаемую целевую систему.

Метод содержит дисциплину и набор практик, чтобы выразить технологию работы команды для достижения цели метода.

Практики, которые составляют (articulate, make up, contained in) метод, должны удовлетворять свойствам:

- однородности (coherency): достижение цели каждой практики даёт вклад в достижение цели всего метода.
- связности (consistency): каждый вход (entry) и результат (result) взаимосвязаны и используются.
- полноты (completeness): достижение целей всех практик означает достижение цели метода и достигает ожидаемого результата.

Эти свойства в начале создания (authoring) метода почти никогда не присутствуют, любой проект начинает набирать практики своего метода по мере понимания ситуации.

Кроме цели, составляющих метод практик и используемой дисциплины в методе ничего больше нет.

Методов десятки тысяч, но практик в них много меньше. Практики как слова, из которых можно составить множество методов-предложений. Каждый проект должен получить свой метод, ни один метод не работает для разных проектов. Так, если проект небольшой, то работа с требованиями может проходить с использованием практики user story, если проект большой, то необходимо уже использовать практику Use Case. Если нужно вести какие-то списки, то "в ворде" можно справиться с текстовым списком не более чем на 400 элементов, а если элементов списка больше, то для его ведения нужно уже использовать какие-то другие практики — другие технологии (базу данных, например), хотя дисциплина ведения списка может остаться той же самой. В зависимости от природы системы, наличия специалистов и доступности технологий используемые даже в похожих проектах методы могут разительно различаться из-за вариации в использованных практиках.

В зарубежной практике слова method, methodology обычно считаются синонимами. По-русски описание метода иногда называется "методология" (не путать с "методикой", которая только часть методологии и означает обычно описание практики).

### Методологическая действительность и действительность предпринятия

Когда мы обсуждаем деятельность, дисциплины, практики — это обсуждение "вообще", не относимое к конкретному проекту, начинанию (предпринятию). Говорят о "методологической действительности/реальности" (не будем отвлекаться на тонкие философские нюансы отличия действительности от реальности), в которой идёт это "обсуждение вообще". Мы описываем деятельность, но пока ничего не делаем. Мы описываем, как оно бывает "вообще", говорим о типах и классах предметов, но не о самих предметах из реальной жизни. Эти типы и классы предметов в тачку не положишь, ногой не ударишь — это абстрактные объекты.

В "реальности предпринятия" обсуждаются уже конкретные объекты и дела этого предпринятия (так называемые "индивиды", individual — которые можно ударить ногой, или погрузить в тачку, т.е. которые имеют протяжённость в пространстве-времени).

Так что какая-нибудь "сборка газодинамического подшипника" может быть описана два раза: в методологической действительности (как вообще собирают такие подшипники) и в действительности предпринятия (как собирается конкретный подшипник с серийным номером №123456).

## *Семь основных альф инженерного проекта*

### Основы системной инженерии: альфы инженерного проекта

Основы (kernel из OMG Essence) включают в себя семь альф трёх дисциплин. Все эти альфы тесно связаны друг с другом, на диаграмме приведены лишь некоторые основные связи. Нужно чётко понимать, что представление инженерного проекта через эти основные альфы — это существенное огрубление реальности. Но именно это огрубление реальности позволяет из "цветущей сложности" выделить главное, на чём нужно будет сосредоточить мышление — какие-то детали при этом неизбежно потеряются, но ситуация "слона-то я и не заметил" будет встречаться реже. В каждом инженерном проекте минимально нужно отслеживать семь альф в трёх дисциплинах, меньше объектов внимания и дисциплин работы с ними иметь нельзя.

Это отслеживание и работа по изменению всех семи основных альф происходит в течение всего проекта. Когда в проекте происходит "пожар", люди работают по ночам и всё внимание уделяется провальной составляющей проекта, знание этого простого факта — необходимости удержания во внимании всех семи альф на протяжении всего проекта — позволяет уберечься от "глупых ошибок".

### Стейкхолдеры

Никаких инженерных проектов не происходит без как-то связанных с ними людей. Инженерные проекты затрагиваются самыми разными людьми, и инженерные проекты затрагивают самых разных людей. Эти люди могут быть как "одиночками", так и организованными в группы, в том числе организованные в группы с известным им распределением полномочий (организации). Эти люди, группы и организациии, которые затрагивают проект, или которых затрагивает проект, называются стейхколдерами (stakeholders/заинтересованные стороны). Перевод "заинтересованные лица" не так хорош, ибо этот термин закреплён в законодательстве за юридическими лицами и при общении с менеджерами-юристами и экономистами возможна путаница.

Стейкхолдеры — это "действующие лица" как в театре) проекта, а исполнители этих ролей — конкретные люди и организации. Мы назовём это "театральной метафорой", при работе со стейкхолдерами всегда нужно помнить формулировку из театральной программки: "действующие лица и исполнители". Нельзя путать "архитектора" и "Василия Петровича" — так же нельзя, как нельзя путать "принца Гамлета" и исполняющего его роль "Василия Петровича".

Стейкхолдеры условно делятся на "внешних" и "членов команды". Стейкхолдеры дают возможности (opportunity) для проведения инженерного проекта: если проект никого не затрагивает (никому не нужен), то его попросту невозможно делать. Если команда может делать проект, но пользователям он не нужен, то такого проекта не будет — разве что члены команды будут работать бесплатно, и будут исполнителями также и других ролей (инвесторов, владельцев, пользователей, клиентов и т.д.).

Стейкхолдеры требуют согласовать с ними определение системы (прежде всего требования — определение системы как "чёрного ящика", ибо как устроена система внутри интересует отнюдь не всех стейкхолдеров) и используют (стейкхолдеры-пользователи) воплощение системы, ради создания которого и затевается инженерный проект.

Простейший рабочий продукт, отражающий альфу "стейкхолдеры" — это список стейкхолдеров. Из информационных систем со стейкхолдерами работают CRM (customer relationship management).

Специально нет никаких особых дисциплин, которые позволяют работать со стейкхолдерами, но можно выделить:

- Конфликтологию (например, метод "принципиальных переговоров" или "гарвардский метод" — найдите в Сети литературу по этому вопросу), чтобы снимать противоречия между требованиями различных стейкхолдеров.
- Коммуникации (communications) для налаживания продуктивного диалога со стейкхолдерами
- Особые техники представления стейкхолдеров (например, "метод персонажа" из книги Алана Купера "Психбольница в руках пациентов" — http://rutracker.org/forum/viewtopic.php?t=1227489, который обобщается с пользователей на любых других стейкхолдеров — http://praxos.ru/index.php/%D0%98%D0%B4%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%D0%90%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B5%D1%80).

### Возможности

Сам инженерный проект начинается с появления возможностей (opportunity) по его проведению — это обстоятельства, которые делают возможным разработку (или доработку — изменение уже имеющейся) системы. Наличие возможности существенно зависит от времени ("окно возможностей" — период времени, в течение которого существует возможность выполнения проекта).

Возможности прежде всего характеризуют пользовательские потребности (пользовательские нужды, user needs — то, что хотят пользователи такого, для чего им поможет наличие воплощения системы), но они также отражают и наличие возможностей команды с развёрнутыми для этой команды технологиями и доступными финансовыми ресурсами в удовлетворении этих потребностей.

Именно возможности мотивируют стейкхолдеров заниматься инженерным проектом, именно возможности объединяют стейкхолдеров на цели выполнения инженерного проекта по созданию целевой системы.

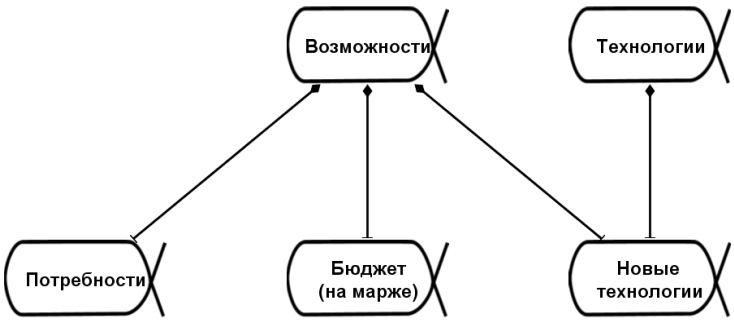
Эти возможности описываются целым рядом возможных рабочих продуктов: "бизнес-планом", "концепцией", "интервью пользователей", "обоснованием инвестиций" и т.д. — в этих рабочих продуктах обосновывается польза разным стейкхолдерам от выполнения инженерного проекта, ибо если нет обоснованных возможностей, то выполнение инженерный проекта не приносит пользы, а приносит вред (например, убытки для инженерной компании).

Из задействуемых для изменений в состоянии возможностей дисциплин нужно указать прежде всего:

- Маркетинг и продажи, стратегирование и предпринимательство — для установления user needs;
- Управленческий (финансовый) учёт — для обоснования прибыльности.

Конечно, в возможностях важны не только нужды/потребности/ожидания пользователей (user needs), но и нужды/потребности/ожидания и остальных стейкхолдеров. Как вы помните, успешная система (точнее, воплощение системы) — это та, которая реализует возможности, т.е. отвечает нуждам/потребностям/ожиданиям стейкхолдеров инженерного проекта.

Вот пример подальф возможностей (обратите внимание, что новые технологии, которые необходимы предприятию, чтобы удовлетворить потребности своих клиентов, одновременно являются подальфами технологий):




Определение системы

Перед тем, как сделать любую систему, её нужно определить (define), ибо нельзя сделать то, что "неопределено" (задача "пойди туда, не знаю куда, найди то, не знаю что" — это больше ведь исследовательская задача, а не инженерная. Чтобы воплотить в нашем четырёхмерном пространстве-времени какую-то систему, нужно как минимум иметь представление об этой системе, "определить" её). Альфа "определение системы" (system definition) служит как раз для этой цели.

У альфы определения системы (system definition) главные подальфы (части) прежде всего:

- Требования (описание назначения системы в её операционном окружении. Требования определяют систему как "чёрный ящик")
- Архитектура (набор ключевых инженерных решений/decisions по тому, как будет устроена система — описание "прозрачного ящика". Изменение каждого из архитектурных решений на поздних стадиях проектирования

ведёт к существенному перепроектированию всей системы).

- Неархитектурная часть проекта (все остальные решения/decisions, т.е. изменение которых на альтернативные не приводит к существенному перепроектированию всей системы)

С определением системы работает прежде всего наука: если какая-то часть системы (или аспект системы) определены, то это означает, что выбран соответствующий метод описания. Наука — это как раз про создание методов описания. Научное знание позволяет определять системы, описывать их в рабочих продуктах — описаниях (descriptions). Но, конечно, определять и описывать системы можно и на основе эвристик, не прибегая к научному знанию. Более того, переход от определения системы (идеального объекта) к описанию системы (материальному объекту) возможен с использованием нотационной инженерии — т.е. для записи определения системы как "мыслей по поводу системы, свойств системы" можно создать инженерный артефакт: нотацию.

Итого: определение системы — это про биты, про информацию, про то, как мы говорим о системе.

Основные дисциплины работы над определением системы — это инженерия требований, проектирование и конструирование (включающие работу с архитектурой системы).

Альфой определения системы занимается системный инженер. Основной рабочий продукт – это описание системы, чаще известный как «проект системы» (design), часто бьётся на множество отдельных документов, баз данных, презентаций, докладных записок, цитируемых стандартов и даже физических макетов.

### Воплощение системы

Воплощение системы (system realization, буквально: вынос в реальность) — это четырёхмерное воплощение системы в нашем материалом мире, это про организованные в пространстве-времени хитрым образом вещества и поля, атомы (а не биты!). Это не про информацию о системе, это сама система. Конечно, воплощение системы будет называться везде по-разному:

- У конструкторов это чаще всего "изделие" или "продукт" (product)
- У проектантов это часто "установка" (plant)
- У проектантов очень крупных объектов (например, атомных станций) это "сложный инженерный объект"

Мы принимаем, что когда мы пишем название системы ("насос"), то мы имеем тут ввиду сам насос как он есть в реальном мире, а не описание насоса (рабочий продукт определения системы).

Пользователям прежде всего нужно воплощение системы, для получения воплощения системы и создаётся инженерный проект.

Очень часто говорят об инженерном проекте по созданию сервиса — например, "сервис стрижки волос". Но это не должно смущать: на самом деле это не проект по созданию сервиса, а проект по созданию системы, оказывающей сервис, например, "парикмахерская", которая будет оказывать "сервис стрижки волос".

Воплощение системы в материальном мире есть и у программной системы: программа ведь не существует без носителя. Но в конкретном случае исполняемая

программа в машинном коде, взятая в какой-то момент её существования подразумевает конкретное сочетание вещества и полей (магнитных доменов в флеш-памяти, заряженных ячеек в оперативной памяти, в регистрах процессора) — каждому такту выполнения программы соответствует какое-то определённое состояние вещества, как и каждому такту работы какой-то другой "железной" системы. Конечно, при рассуждениях происходит абстрагирование от этой "физикализации" программы, но полезно помнить, что исходный код — это рабочий продукт определения системы. Одно определение системы обычно может пригодиться при создании тысяч и тысяч воплощений системы. Так и в случае софта: воплощение программной системы — это исполняющаяся (иногда в тысячах и миллионах экземплярах) программа, а не исходный её код или даже "исполняемая программа" в машинных кодах, но на которую ещё не передано управление. И инженеры-программисты занимаются определением системы, а вот воплощением занимаются часто люди из совсех других подразделений и организаций (сисадмины, операторы). Попытка обратить внимание программистов на важность воплощения программной системы сегодня обсуждается как проблема DevOps (developers-operators).

Воплощение системы используется стейкхолдерами, оно реализует возможности. Воплощение системы удовлетворяет определению системы.

Основная дисциплина работы над воплощением системы — это производство (изготовление отдельных частей, сборка их, испытания).

Альфой воплощения системы занимается системный инженер.

### Команда

Команда (team) инженерного проекта — это не просто какие-то люди или организации (группы людей с оборудованием и известным распределением полномочий). Это люди с совершенно определённым набором компетенций, нужных для реализации проекта, при этом речь идёт не только об инженерных, но и о менеджерских и других прикладных компетенциях.

Команда создаёт определение и воплощение системы, команда выполняет работы, команда применяет практики (дисциплины в головах и технологии в предпринятии).

Команда должна работать как слаженный коллектив, для этого её нужно организовать из отдельных составляющих её людей. Для того, чтобы каждый человек занял своё место на логистическом "конвейере" (ибо если какие-то места на этом "конвейере" не будут заполнены людьми, то целевая система просто не сможет выпуститься — необходимые на этом рабочем месте работы не будут произведены), нужно его "уговорить". Это функция "комиссара", пропагандиста, специалиста по менеджерской дисциплине "лидерство" (leadership). Лидер выполняет работы, которые можно описать двояко (помним, что это два разных описания одной и той же деятельности):

- Убалтывает исполнителей ролей команды играть в этой команде необходимые роли (убалтывает путать "личное" и "общественное").
- Осмысляет жизнь исполнителей ролей тем, что они приносят стейкхолдерам пользу, их жизнь и работа имеют значение для окружающих.

### Работы

Для того, чтобы инженерный проект был успешен, команде проекта нужно провести

работы (works) — и отслеживать состояние этих работ в ходе всего инженерного проекта.

Конечно, содержание этих работ определяется каждым из членов команды проекта — но есть особая работа по проведению работ (operations management), это работа операционного менеджера. Прежде всего, требуемые работы нужно:

- Учитывать во всём их содержательном разнообразии, чтобы ответить на вопрос “что делать”.
- Планировать (schedule), т.е. предлагать распределение этих работ во времени и назначать этих работы исполнителям.
- Определять достаточность ресурсов и контролировать выполнение плана работ для понимания того, вовремя ли работы будут закончены (т.е. не закроется ли окно возможностей раньше того момента, когда эти возможности будут удовлетворены результатом проекта).

С точки зрения операционного менеджера вся организация представляет собой набор рабочих мест/станций, на которых требуемые проектом различные ресурсы (люди, инструменты, расходные материалы) задействованы для выполнения содержательных работ, а также продукты работ движутся между этими рабочими местами/станциями.

У того, кто занимается работой, мышление представляет проект как некоторую трубопроводную сеть, по которой текут (flow, но по-русски тут будет “идут”, “проходят”) работы, материалы, люди, информация так, что из “входного” информационного, человеческого, материального сырья получаются “выходные” воплощения системы — а обратным ходом текут/идут/проходят вырученные за воплощения системы деньги. Это логистическое, операционное мышление.

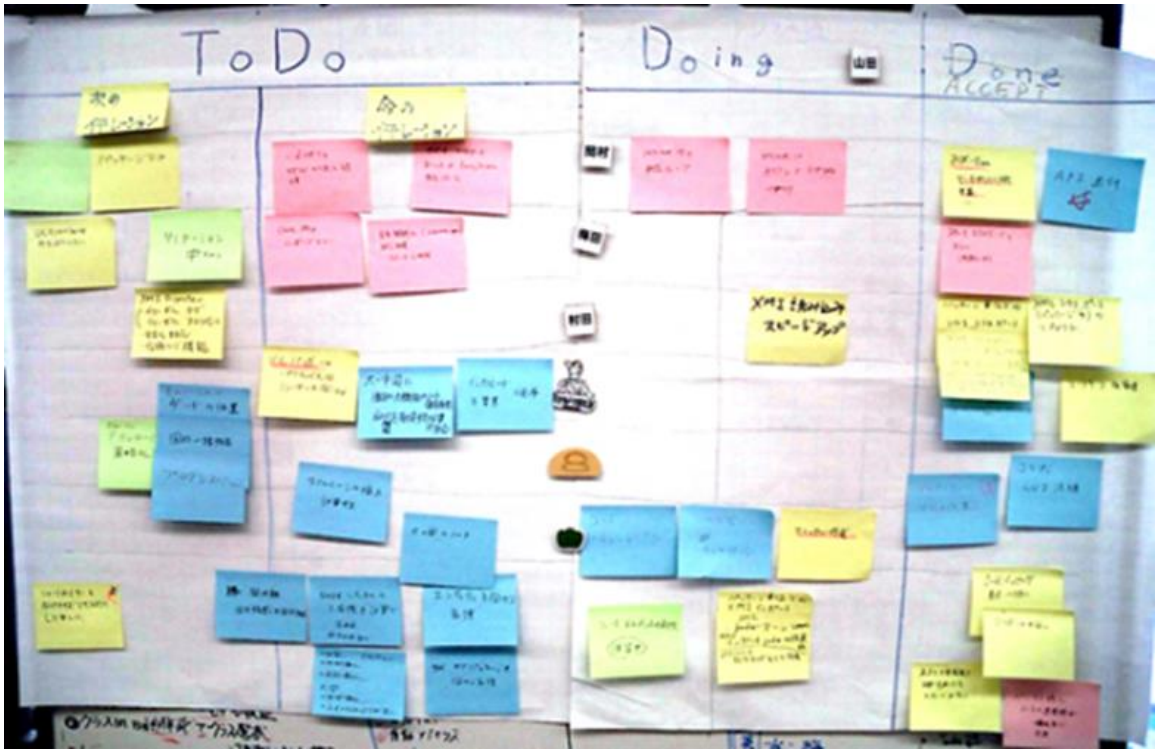
Из дисциплин, которые работают над альфой “работа”, можно указать:

- Операционный менеджмент (из Lingvo: operations management — управление операциями [производством]. Управление производственным процессом фирмы, в отличие от стратегического менеджмента, управления персоналом и других составных частей управления организацией; исторически первое название этой деятельности production management было изменено на operations management, т.к. по сути "производство" существует практически во всех организациях, и в том числе в сфере услуг, страховании, банковском деле и т. д., а слово production ассоциируется лишь с материальным производством). На английском языке общепринятое определение проще — Operations Management is the process by which an organization converts inputs (e.g. labor, material, knowledge, equipment) into outputs (goods and services). На русском языке наиболее часто используется до сих пор старая форма "управление производством" и много реже "управление операциями" или прямая калька "операционный менеджмент". “Исследование операций” и даже “теория массового обслуживания” также довольно частые термины, хотя под ними чаще имеется ввиду использование специального математического аппарата для расчёта времени работы.
- управление цепочками поставок (supply chain management)
- управление проектами (project management), управление процессами (process management), ведение дел (case management)
- планирование и управление производством (planning and production

management)

- логистика (logistics)
- операционная стратегия (operation strategy)
- управление сервисными операциями (management of service operations)
- улучшение производительности (performance improvement)
- планирование ресурсов предприятия (enterprise resource planning) и управление ресурсами (resource management)
- get things done (GTD) — система персонального планирования, "как доводить дела до конца"

Вот один из видов рабочих продуктов, отражающих альфу "работы" (это простейший issue tracker):




Технология

Дисциплина — она в головах членов команды. Но в организации есть технология: поддержанный необходимыми рабочими продуктами и инструментами способ работы (way of working). Практика = дисциплина+технология (метод = полный набор дисциплин и технологий для выполнения какой-то работы).

Технология существенно зависит от состава выполняемых работ (технология пошива обуви не нужна при проектировании медицинской аппаратуры анализа крови, и наоборот), технология требуется для команды (компетенция проектирования в 2014 году не может быть реализована без компьютеров с установленными на них информационными системами САПР — системами автоматизации проектирования, системами инженерных мультифизических расчётов, системами управления жизненным циклом PLM/product lifecycle management). Бессмысленно привлекать в команду человека и одновременно не обеспечивать его технологией, и не давать фронта работ: все альфы предпринятия тесно зависят друг от друга. Если есть работа "копать траншею длиной 500 метров",

то нужно озаботиться не только нужным количеством землекопов или экскаваторщиков, но и лопатами или экскаваторами. Этот пример также показывает, что для каждой работы могут быть использованы самые разные технологии, и тем самым выполнение инженерного проекта включает выбор (а иногда — выбор, закупку и разворачивание) для его выполнения технологий.

Управление технологиями (каждая из которых имеет свои плюсы и минусы и требует для своего использования людей в команде с разными компетенциями) это отдельная дисциплина инженерного менеджмента.

Удивительно, но люди часто не задумываются о тех технологиях, которые они используют. Что будет, если бригаде землекопов дать экскаватор?


Они потратят целый день на то, чтобы отвинтить "лопату" от экскаватора, а потом попробуют организовать бригаду лопатодержателей, ибо одному человеку трудно будет управляться с такой большой лопатой! Ну, и затребуют пару сотен килограмм изоленты: обмотать неудобную ручку у этой огромной лопаты. А остальное (кабину, двигатель, систему тросов, гусеницы) выбросят: землекопы не знают, для чего все эти лишние детали, привинченные к лопате. Так что инструменты поддерживают те или иные дисциплины, а исполнители должны быть компетентны в использовании инструментов и дисциплинированы в своём мышлении.

Упражнение: Какие технологии используются в вашем проекте? Приведите три примера и для каждого дайте пару альтернативных технологий (например, 3D-моделирование с использованием Autodesk Inventor вместо 2D-моделирования в Autodesk AutoCAD или порождающего проектирования/generative design в специально написанном программном средстве).

**Контрольные вопросы**

Опишите для своих пяти последних командных проектов: какими основными альфами вы в них занимались преимущественно?

Опишите, внимание к каким альфам у вас отсутствовало в пяти последних командных проектах, в которых вы участвовали?

По каким альфам вас не учат работать профессионально в ВУЗе? По каким учат?

Как вы считаете, какими основными альфами вы будете заниматься прежде всего после окончания ВУЗа? Хватит ли вам знания дисциплин по работе с этими альфами?

# 5. Системный подход

## Понятие “подхода”

“Подход” — это когда разработанные в рамках одной дисциплины приёмы работы (в том числе приёмы мышления) переносят в какие-то другие области.

Очень часто “подход” является синонимом “практики” или даже “метода” (помним, что в разных речевых сообществах слова “практика” и “метод” отнюдь не используются в точных терминологических значениях, в отличие от нашего курса — тем не менее, общее сходство в их значениях несомненно). Слово “подход” означает обычно, что какие-то “практики” или даже целый “метод” работы были отработаны в какой-то одной предметной области, отрефлектированы (т.е. явно описаны в отрыве от предметной области, на объектах которой они отрабатывались), а затем перенесены туда, где раньше они не использовались.

Так что использование слов “подход” или “метод” по большей части ситуативно: научный подход вы используете при описании какого-то фрагмента реальности или научный метод — это не так важно. Все поисковые системы подразумевают синонимичность подхода (approach) и метода (method). Если вы будете искать “научный подход”, то вам покажут “научный метод”: перенос методологического знания из одной предметной области в другую по мере развития ситуационной инженерии методов становится обыденным.

Тем самым и саму системную инженерию можно считать “подходом”, если мы придём с её практиками и методами в те области, в которых она раньше не использовалась — например, будем использовать системную инженерию при создании искусственных живых организмов или наноботов. Ну, или “подход системной инженерии” уместно говорить в ситуациях, когда мы пытаемся от советских инженерных методов перейти к методам системной инженерии.

В самой системной инженерии используется “системный подход”, и можно найти множество других “подходов”, например, архитектурный подход (где методы архитектурной работы распространились из традиционной архитектуры на работу со сложными техническими системами и даже программными системами).

Упражнение: какие подходы вы знаете?

## Системный подход в системной инженерии

Системный подход — это когда наработанное где-то (в данном случае уже неважно где) системное мышление переносится в другие дисциплины — например, в (системную) инженерию.

Есть много легенд, почему системная инженерия взяла на вооружение системный подход. Вот одна из них в вольном пересказе:

> Когда два суперсложных проекта 20 века попытались объединить — речь идёт об американских “Манхэттенском проекте” создания атомной бомбы и проекте разработки баллистических ракет как средства доставки получаемых бомб — совокупная сложность проекта, подразумевающая учёт в одном проекте результатов работы множества дисциплин, перестала умещаться в одной голове “генерального конструктора”. В те далёкие времена, когда самолёты назывались по имени генерального конструктора-гения (все эти “мессершмиты” и “ильюшины”), встретились задачи, которые по сложности выходили за возможности конкретного

> гения овладеть множеством разных дисциплин, и нужно было выработать какой-то отчуждаемый от гениального человека способ совладания с этой сложностью. Тогда вспомнили, что инженеры-иммигранты с заводов "Мерседес-Бенц" используют для своей работы "системный подход", который они позаимствовали у биологов, изучавших биогеоценозы — сложнейшие биологические системы, затрагивающие уровень сложности выше, чем сложность отдельного организма. Этот "системный подход" — использование мышления в терминах систем — появился в инженерном деле и после этого уже было не сложно наращивать сложность разрабатываемых систем, не опасаясь выхода этой сложности за пределы одной гениальной головы.

Тут обычно задаётся вопрос: как у нас в СССР, без всего этого системного подхода работали атомные станции и прочие сверхсложные объекты? А вы не спрашивайте о том, сколько людей было занято, каковы были их условия труда (достаточно вспомнить лагеря-шарашки), сколько было человеческих жизней загублено в этих проектах по совокупности причин, какое качество было создаваемых систем, сколько эти системы стоили (особенно в ситуации, когда реальной стоимости посчитать никто не мог в силу неконвертируемости валюты). Ну, также вопрос о частоте обновления продуктной линейки и смены технологий. Да, в 50-х годах инженерные технологии СССР и Запада были вполне сравнимы (с учётом огромных послевоенных заимствований), в 60-х они начали стремительно расходиться в качестве и количестве, в 70-х разница была уже очень заметна, а в 80-х это было отставание "навсегда", которое до сих пор не преодолено.

Конечно, "если долго мучиться, что-нибудь получится" — при неограниченных средствах и времени всегда можно создать великолепную систему, переделывая её заново при каждой ошибке, выправляя каждый вновь найденный дефект. Как сделать так, чтобы получалось не "что-нибудь" и не нужно было мучиться? Нужно использовать системный подход, позволяющий применить "разделение интересов" (separation of concerns) на практике. Недаром в системной инженерии есть поговорка "с первого раза правильно": если речь идёт о междисциплинарной работе, то каждая из дисциплин видит свои ошибки и даёт свои решения. В результирующем проекте этих ошибок тем самым будет меньше, а решения будут более качественными.

Часто забывают, чему противопоставлялся системный подход в момент его появления, и почему так много внимания уделяется тем якобы "банальностям" и якобы "здравому смыслу", которые в совокупности и составляют системный подход. Но это сейчас в 21 веке многие положения системного подхода выглядят вполне естественными, а ведь ещё в середине 20 века это было далеко не так.

Системный подход с его вниманием не только к частям, но и целому (холизм) противопоставляется прежде всего редукционистскому подходу. В редукционистском подходе (часто неявно) утверждается, что мы должны дать детальное "научное" (то есть в рамках определённого научного предмета) описание любого объекта, просто повышая уровень его детальности при любых встречающихся затруднениях, сводя изучение целого к изучению его отдельных частей. Но почему представители системного мышления называют это "редукционистским" подходом? Потому как доктрина редукционизма полагала, что любое "высшее проявление" можно свести к "низшему, в частях системы", если постараться. Бег зайца можно свести к химическим реакциям в молекулах зайца, то есть заяц сводим к его химии. Или атомная электростанция сводима к набору атомов

физических элементов, которые нужно только собрать в правильные места в пространстве. Согласно редукционизму, инженерия вся сводится к правильному использованию физики, и только — ибо инженерный объект это физический объект, и только. Никаких "систем", "жизненных циклов", "требований" и "архитектур", только законы физики!

Системный подход, в отличие от редукционистского утверждает, что "сводимости" одних дисциплин к другим нет, мир нужно описывать мультидисциплинарно, "полинаучно" — зайца одним образом, его клетки другим образом, молекулы в клетках третьим образом, а молекулярные орбитали в атомах этих молекул четвёртым образом. Главное в системах — это эмерджентность (emergence, http://en.wikipedia.org/wiki/Emergence) — то, что из простых взаимодействий каких-то частей появляется что-то, что абсолютно никаким образом не содержится в каждой из частей. Например, ни одна часть часов не содержит "время" или "измерение времени". Это свойство появляется (emerge/возникает) только в результате взаимодействия всех частей, свойство механических часов измерять время нельзя сводить к свойствам отдельных их частей: пружин, шестерёнок, храповиков. Ключ к пониманию системы не в её частях, а в том новом, что появляется при их взаимодействии.

И эти теории (компактные описания "как устроено") разнятся не только на разных уровнях отношения "часть-целое" (часто при этом говорят о "метасистемном переходе" — когда человека или атомную станцию перестают рассматривать как набор молекул с их взаимодействиями на химическом уровне), но и даже для одного уровня (когда один и тот же уровень рассмотрения по отношению часть-целое обеспечивается целым рядом дисциплин — когда атомной электростанцией занимается инженер-механик, инженер-теплотехник, инженер-электрик, инженер по безопасности и т.д.).

Системный подход сразу оговаривает многодисциплинарность (в отличие от монодисциплинарности редукционистского подхода — "наша дисциплина объяснит всё многообразие явлений") рассмотрения системы. Каждая дисциплина привносит свою теорию, свой компактный набор описаний мира, пригодный для ответов на свои вопросы. Системный подход позволяет сразу к этому приготовиться: с системным подходом жить не легче, чем с редукционистским (меднолобым фанатам одной идеи ведь жить всегда легче, так?), зато можно добиться лучших результатов, применяя различные теории там, где они могут быть применены, и не применяя там, где их применять нельзя (ибо в каждой теории оговаривается тот круг явлений, к которым эта теория в принципе может быть применена).

Системное мышление (system thinking) — это приложение системного подхода к решению практических задач. Так, системная инженерия — это приложение системного подхода к решению инженерных задач. Этот вариант системного мышления мы будем называть системноинженерным мышлением.

Есть и другие варианты системного мышления, ибо существует множество разновидностей системного подхода, значительное число этих разновидностей посвящено попыткам разбирательства с системами из людей (см., например, обзоры http://www.situation.ru/app/j_art_1052.htm и http://rudocs.exdat.com/docs/index-421147.html?page=8 — при этом помним, что systems engineering до середины 80-х годов по-русски переводили словом "системотехника", которое и использовано в этих обзорах). Сегодня системная инженерия представляет собой одну из самых бурно развивающихся ветвей системного движения, при этом она активно впитывает и идеи других направлений системного движения.

Варианты системного подхода

Есть огромное число вариантов системного подхода, каждый из них по-своему определяет систему и развивается на базе разных областей деятельности:

- Биологии. Например, http://en.wikipedia.org/wiki/Systems_biology — Systems biology is a biology-based inter-disciplinary field of study that focuses on complex interactions within biological systems, using a holistic approach (holism instead of the more traditional reductionism) to biological and biomedical research. Particularly from year 2000 onwards, the concept has been used widely in the biosciences in a variety of contexts. One of the outreaching aims of systems biology is to model and discover emergent properties, properties of cells, tissues and organisms functioning as a system whose theoretical description is only possible using techniques which fall under the remit of systems biology. These typically involve metabolic networks or cell signaling networks. Systems biology makes heavy use of mathematical and computational models. Обратите внимание на традиционные для системного подхода противопоставление редукционизму, внимание к эмерджентности, использованию мат.моделей.
- Экологии (http://en.wikipedia.org/wiki/Ecology — помним, что одно из основных понятий экологии — это "экосистема", а принцип — тот же противопоставляемый редукционизму холизм).
- Менеджменте и теории организации (довольно старый обзор, но на русском языке см. В статье М.С.Джексона "Системному мышлению в менеджменте — 50 лет" — http://cyberleninka.ru/article/n/sistemnomu-myshleniyu-v-menedzhmente-pyatdesyat-let). Учитывая то, что системная инженерия всё больше и больше от создания успешных физических систем переходит к созданию успешных киберфизических систем, а современный тренд в ней — переход к созданию киберфизикосоциальных (cyber-physic-human) или иначе социотехнических (sociotechnical) систем, внимание системных инженеров к этому направлению системного движения сейчас очень велико. Кстати, обратите внимание, что слово governance по традиции в статье переводится как "управление". Пожалуй, его более точно переводить как "контроль" (в смысле "невозможности вырваться из-под контроля"). Так, corporate governance это про то, как организацию держат подотчётной собственникам, и менеджерам и сотрудникам не дают самоуправничать в ней. Поэтому "theory of governance" на первой странице статьи по ссылке — это про теорию "удержания под контролем собственника".
- Исследование операций (теория массового обслуживания, производственное планирование и т.д. — http://en.wikipedia.org/wiki/Operations_research). Когда-то в классической книге Л.Берталанфи по общей теории систем он определил в качестве прикладных областей использования системного подхода (общей теории систем в широком смысле) системную инженерию, исследование операций и инженерную психологию. Исследование операций по факту это часть менеджмента (иногда говорят, что использование идей исследования операций это "научный менеджмент"), но должна быть очевидна связь системного движения и этой дисциплины.
- Системная инженерия — приложение идей системного подхода (использование системного мышления) в инженерной деятельности.
- Множество других использований. Постепенно системный подход становится основным способом представления знаний о мире. То, что мир состоит из

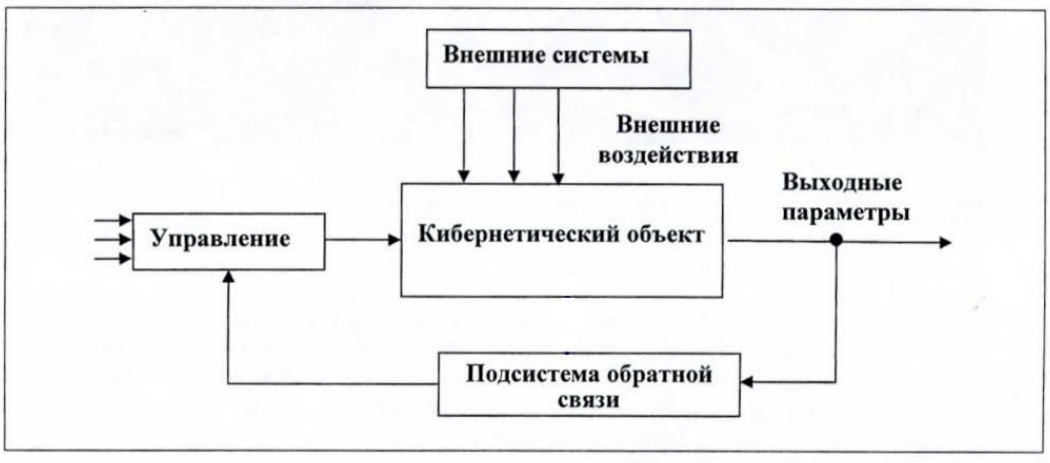
систем, становится общепринятым, и поэтому иногда говорят про системный подход как современную онтологию (онтология — это как раз ответ на вопрос “каков мир? Что в нём есть?”. Ответ системного подхода — мир системен, в нём есть системы).

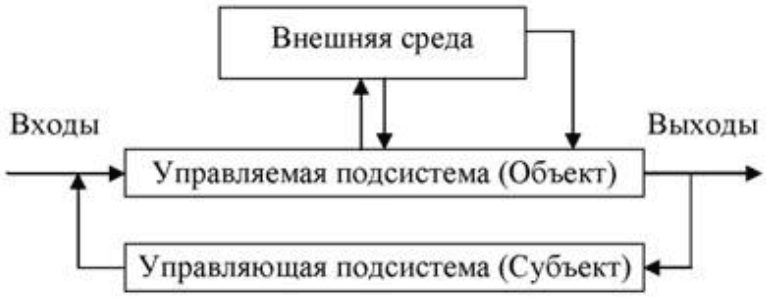
Развитие системного подхода проходит в рамках так называемого “системного движения”. От другого рода организованностей (например, научных школ, профессиональных сообществ, научных дисциплин и т.д.) “движение” (movement) отличается отсутствием координационного центра при полной автономии входящих в него групп, разнообразием форм организации и деятельности, отсутствием какого-то явного механизма согласования целей и форм обмена опыта. Тем не менее, при полной независимости участников движения друг от друга, свободе и автономии в выборе ими целей и методов работы, все участники движения обнаруживают некоторую общность (в нашем случае — общность использования системного подхода для задач в своих предметных областях, развитие системного мышления на материале своих предметных областей).

### Системный подход и кибернетика

Кибернетика (cybernetics) — это придуманный Нобертом Винером в 1948 году междисциплинарный подход к научному изучению управления/контроля в животных и машине ("the scientific study of control and communication in the animal and the machine." — http://en.wikipedia.org/wiki/Cybernetics). Часто кибернетику определяют как науку об управлении в технике и природе. Сам Норберт Винер в своих поздних работах писал, что погорячился с кибернетикой, особенно в той части, в которой писал про живых существ — и жизнь показала, что в своих поздних работах он был прав, кибернетика сегодня из мейнстрима стала маргинальным подходом, при этом осталась только “техническая кибернетика” (о живой природе уже и речи нет). Тем не менее, кибернетика в 50-70-е годы 20 века оказала огромное влияние на системное движение.

В кибернетическом подходе среда (environment, состоящая из внешних систем) влияет на целевую систему, состоящую из управляемой подсистемы и управляющей подсистемы, обеспечивающей обратную связь (feedback) на управляющие воздействия. Так, с точки зрения кибернетики паровая машина состоит из собственно паровой машины и регулятора Уатта, обеспечивавшего обратную связь. Вот разные варианты кибернетической схемы, в соответствии с которой нужно было рассматривать системы:






Это нехитрое соображение про "обратную связь" оказалось очень продуктивным: во всех системах начали выделять управляемую и управляющую компоненту, рассматривать самые разные виды обратной связи.

На какой-то момент кибернетика и системный подход начали существенно путаться друг с другом, а в нашей стране это сыграло злую шутку: системная инженерия (systems engineering), переведённая как "системотехника" развивалась главным образом в организациях, занимавшихся задачами управления, понимаемых как использование компьютерных управляющих систем— и вместо ракет, подводных лодок, самолётов, автомобилей, медицинской аппаратуры и других традиционных "железных" объектов системотехника свелась к занятиям именно управляющими компьютерными системами с использованием разнообразных математических моделей.

Вместе с тем из кибернетики вышли множество интереснейших школ, например (но не ограничиваясь ими):

- теория автоматического регулирования (рассматривающая, какое нужно выдавать управляющее воздействие по получению обратной связи — линейное, нелинейное, упреждающее и т.д.: http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F. Обратите внимание, что статья в википедии называется не теория автоматического регулирования, а теория автоматического управления. Обратите внимание, как похожа диаграмма типовой схемы системы автоматического управления на типовые схемы кибернетической системы. Посмотрите также на учебную программу курса теории автоматического регулирования, обратите внимание на годы издания литературы: http://www.inp.nsk.su/students/radio/courses/teor_auto/index.shtml. Сравните с http://en.wikipedia.org/wiki/Control_theory, обратите внимание на использование терминов feedback systems, feedback control, control loops).
- Системная динамика (system dynamics) как более-менее "всеохватное моделирование". В системной динамике можно представлять какие-то объекты и моделировать их воздействия друг на друга, включая обратные связи. Это очень простой подход: объекты моделируются "резервуарами", а прямые и обратные связи между ними — трубопроводами. Для системной динамики есть очень удобные и мощные компьютерные программы, позволяющие автоматически строить и решать системы уравнений для таких моделей (http://en.wikipedia.org/wiki/System_dynamics). В настоящее время системная динамика постепенно вытесняется другими методами

моделирования (например, акаузальным объект-ориентированным языком Modelica — http://modelica.org).

Нужно понимать, что системный подход и кибернетика — ближайшие родственники, но они не сводимы друг ко другу, понятия "обратной связи" и управления (как в варианте control, так и в варианте governance) центральные в кибернетике, но отнюдь не центральные в системном подходе как таковом.

В современной системной инженерии слово "кибер" (cyber) указывает на наличие в системе компьютера, но не означает использования кибернетики. Так, "киберфизическая система" означает, что в составе системы есть компьютер (или множество компьютеров) — но этот компьютер вовсе не обязательно работает, обеспечивая "обратные связи". Функции компьютера в современных системах многообразны и не сводятся к отработке обратных связей. Так, компьютер может проводить диагностику неполадок в системе, или обеспечивать удобство управления системой для человека-оператора, или записывать информацию по тем режимам, в которых работала система ("чёрный ящик"): этим всем занимается программная инженерия и системная инженерия, а не кибернетика.

### Сложность и меры сложности

Понятие сложности интенсивно разрабатывалось в рамках системного подхода, но окончательного согласия по этому поводу нет. Так, Seth Lloyd собрал различные определения для мер сложности (http://web.mit.edu/esd.83/www/notebook/Complexity.PDF). Все эти определения обычно относятся к попыткам ответа на три вопроса:

1. Как трудно описать систему? Обычно это измеряется в битах, затрачиваемых на представление описания. Мерами сложности тут будут информация, энтропия, алгоритмическая сложность или алгоритмическое содержание информации, максимальная длина описания, информация Фишера (Fisher), энтропия Рени (Rényi), длина кода (беспрефиксного, Хаффмана, Шэннона-Фано, корректирующего ошибки, Хэмминга), информация Чернова, размерность, фрактальная размерность, сложность Lempel-Ziv.

2. Как трудно создать систему? Сложность как трудность создания измеряется во времени, энергии, долларах и т.д. Меры сложности тут вычислительная сложность, временнАя вычислительная сложность, пространственная вычислительная сложность, основанная на информации сложность, логическая глубина (depth), термодинамическая глубина, цена, шифрованность (crypticity).

3. Какая степень организованности? Тут может быть два варианта:

а) "результирующая сложность" (effective complexity), трудность описания организационной структуры, неважно корпоративной, химической, клеточной. Приведём их по-английски: Metric Entropy; Fractal Dimension; Excess Entropy; Stochastic Complexity; Sophistication; Effective Measure Complexity; True Measure Complexity; Topological epsilon-machine size; Conditional Information; Conditional Algorithmic Information Content; Schema length; Ideal Complexity; Hierarchical Complexity; Tree subgraph diversity; Homogeneous Complexity; Grammatical Complexity.

б) количество информации, которой нужно обмениваться между частями системы из-за такой организационной структуры: Algorithmic Mutual Information; Channel Capacity; Correlation; Stored Information; Organization.

Есть и понятия, которые не являются понятиями сложности, но очень близки: Long—Range Order; Self—Organization; Complex Adaptive Systems; Edge of Chaos. Есть и совсем альтернативные меры сложности (например, основанные на скорости описания оцениваемых объектов, а не объёме этого описания — http://people.idsia.ch/~juergen/speedprior.html).

Ситуация с понятием "сложность" очень характерна для системного подхода: употребляемые в нём слова кажутся вполне "бытовыми" и имеющими ясное и интуитивно понятное с детства значения. Но нет, слово вдруг оказывается термином, за которым скрывается очень сложное и противоречивое понятие, с этим понятием работают самые разные логические или количественные модели, проводятся количественные измерения самых разных его характеристик.

В рамках настоящей книги мы будем считать сложной систему из достаточно большого количества элементов, настолько большого, чтобы в одной голове не получалось оценить все связи и взаимодействия (атомная станция, микрочип процессора — десятки миллионов или даже миллиардов индивидуальных объектов). Но вы должны помнить, что кроме этого неформального понимания есть и другие, формальные понимания сложности, есть множество теорий сложности, из которых эти понимания пришли.

*Термин "система"*


Термин (слово) "система" используется минимально в трёх различных смыслах, которые следует различать:


1. Для обозначения понятия "система" из системного подхода (мы тут будем главным образом использовать вариант системного подхода, наиболее типичный для современной системной инженерии). Увы, это не самое частое использование слова. В системном подходе система определяется как иерархия холонов — холархия. Иерархия — это структура из дерева отношений похожих элементов. В холархии как виде иерархии это отношения "часть-целое", а объекты этих отношений называются "холон" (holon) — по предложению Артура Кёстлера. Холон — это что-то, что является одновременно целым для своих частей и само является частью для какого-то объемлющего целого. Система — это холон, у которого есть появляющиеся (emergent) свойства, получающиеся от взаимодействия его частей. "Эмерджентность" — это главное свойство системы: "целое больше, чем сумма его частей". Часы больше, чем сумма их шестерёнок или микросхем. Часы могут показывать время, а шестерёнки или микросхемы времени показывать не могут.

Обязательно нужно добавить, что в системной инженерии холоны представляют собой индивиды, имеющие пространственно-временную протяжённость (extension, "основные свойства любой вещи — длина, ширина, высота: место, занимаемое в пространстве" как сформулировал Декарт, а современная физика добавила к этому протяжённость во времени — системы занимают место в четырёхмерном пространстве-времени). Простой критерий: индивиды (а, следовательно, и системы) можно "пнуть" (kick) или "погрузить в тачку".

Для контраста посмотрите, как обсуждают холон гуманитарии не-инженеры: http://en.wikipedia.org/wiki/Holon_%28philosophy%29.

Дальше мы будем обсуждать главным образом системы из системного подхода в варианте системной инженерии.

2. "Система" из систематики — различные классификационные (таксономические) "системы". Это тоже иерархии, но элементами в них являются классы (множества), отсюда и название — классификаторы. Между классами в классификациях отношения специализации (класс-подкласс: подкласс это специализированный класс). Классифицируются в конечном итоге индивиды, которые связаны с классом

отношением классификации.

Конечно, в системном подходе системы часто классифицируются по самым разным признакам, но нужно помнить, что “система классификации” — это не система из системного подхода, там нет эмерджентности, нет отношений часть-целое и холонов. Класс (множество) центробежных насосов это не часть класса насосов как целого, а специализированный класс насосов. Все классифицированные как насосы индивиды не взаимодействуют между собой, порождая новое свойство. Они просто “сгруппированы в уме”, так как похожи друг на друга (все они — насосы). Классификаторы и таксономии — это предмет систематики, а не системного подхода.

Системы из инженерного системного подхода классифицируются “системами” из систематики.

3. “Система” как указание на какой-то набор правил, процедур, обычаев, имеющий какую-то (совсем необязательно иерархическую) структуру. Тут слово “система” указывает на какую-то упорядоченность, неслучайность, продуманность. Это не имеет отношения к системному подходу, не подразумевает специально устроенного мышления, похожего для всех этих разных систем: мышление, которое порождало все эти якобы системы — это не системное мышление. Хотя и тут опытный глаз сможет уловить какие-то “части-правила” и эмерджентность “целой системы”, демонстрирующей в целом наборе правил что-то большее, чего нет в каждом отдельном правиле. Но нужно помнить, что скорее всего, ни при создании всех этих “систем”, ни при попытках их как-то понять и проанализировать никакого системного мышления не использовалось.

В настоящем курсе системноинженерного мышления мы будем настаивать, что любая система — это материальный (из вещества и полей) объект, и уж точно не набор правил, не абстрактный объект типа “множество”.

## Классификация систем по ISO 15288

Для системного инженера всё вокруг — это системы, но все эти системы не только разные сами по себе, они разные по их роли в инженерном проекте и разные по интересующим стадиям их жизни (говорят о стадиях “жизненного цикла” системы — хотя никакого “цикла” там нет, да и “жизнь железки” тоже ведь не жизнь! Но об этом позже). Для показа этой разницы ISO 15288 вводит следующую классификацию систем, которая неявным образом включает в себя и стадию жизненного цикла рассмотрения системы:

- **Целевая система (system-of-interest)**

- **Система в операционном окружении (system in operational environment)**

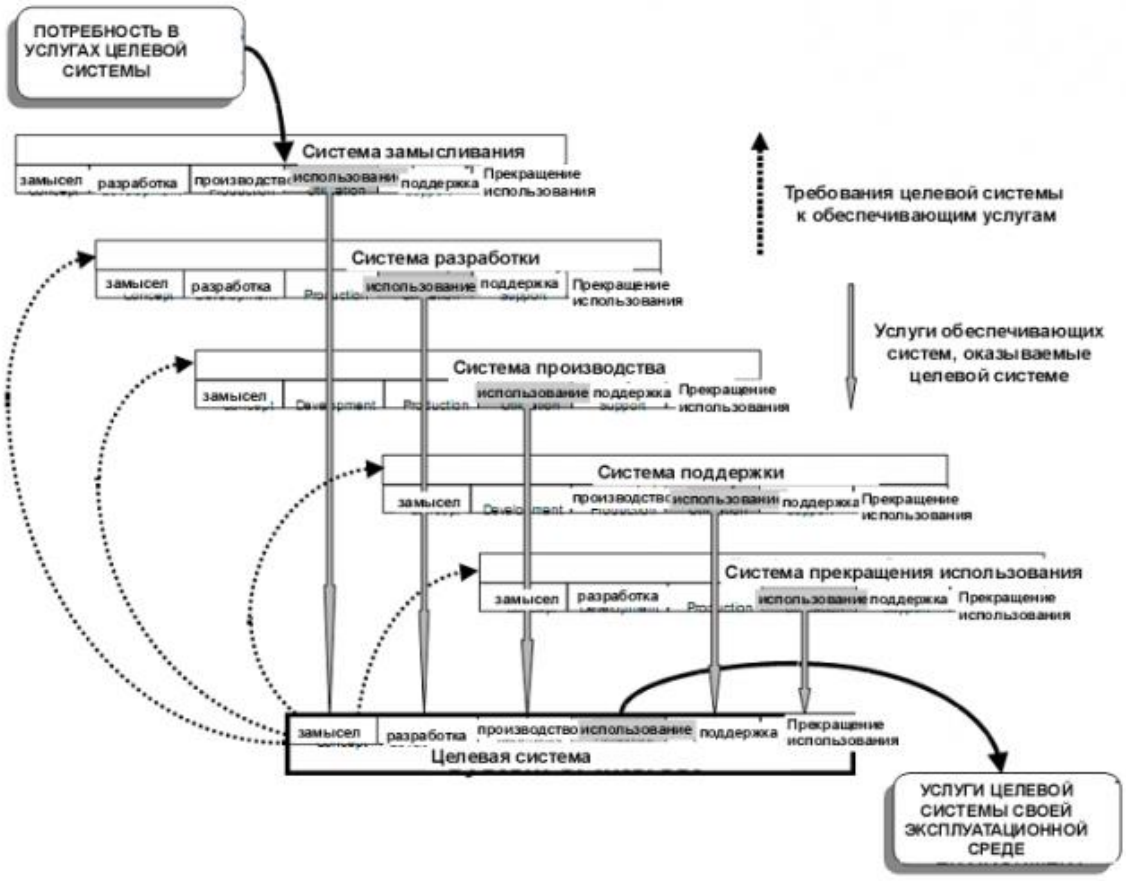
- **Обеспечивающая система (enabling system)**

Целевая система (system-of-interest) — та, которая подлежит созданию (или модернизации) командой инженеров и рассматривается на всём протяжении жизненного цикла. Например, насос.

Система в операционном окружении, система в эксплуатационной среде/операционном окружении (system in operational environment) — одна из систем, которые окружают целевую систему в момент её эксплуатации. Например, трубопроводная система, к которой подключён насос во время эксплуатации.

Обеспечивающая система (enabling systems) — система, которая создаёт и поддерживает систему в ходе её жизненного цикла. Например, цех, который производит насос.

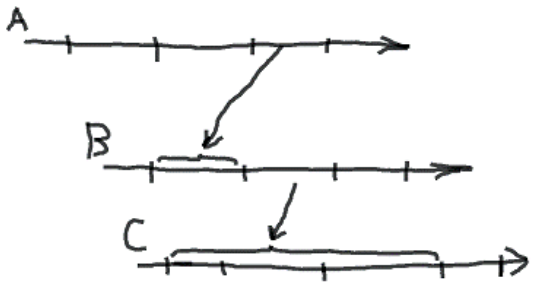
Нужно понимать, что любую систему можно классифицировать либо как целевую, либо как обеспечивающую, либо как систему в операционном окружении. Цех, как обеспечивающую систему, которая производит насос, тоже кто-то проектировал и строил. Инжиниринговая компания, которая проектировала и строила цех, тоже была кем-то создана — и она тоже обеспечивающая система по отношению к цеху. Вот диаграмма, показывающая, что существует множество обеспечивающих систем, которые на стадии своей эксплуатации (operation, использования) выполняют работы по обеспечению (enabling) той или иной стадии жизненного цикла целевой системы. Сама же целевая система на стадии своей эксплуатации работает в составе систем своего операционного окружения, выполняя свою функцию:




Нужно всегда понимать, о какой системе вы говорите: когда вы говорите “топор”, то непонятно — вы делаете топор (целевая ваша система), вы используете топор для колки дров (целевая система — дрова, топор — одна из обеспечивающих систем, необходимая для подготовки целевой системы к эксплуатации — сгоранию в печке) или топор для вас одна из систем в операционном окружении целевой для вас колоды, совместно с которой топор должен колоть дрова и свойства которого

вы выясняете для того, чтобы спроектировать и изготовить правильную колоду (http://forum.rmnt.ru/threads/koloda-dlja-kolki-drov-nou-xau.102202/).

Есть более простой способ показывать диаграммы жизненного цикла обеспечивающих систем: жизненный цикл на них рисуется изображающими время стрелками с зарубками, отделяющими стадии жизненного цикла, а отрезки работы обеспечивающей системы над соответствующей целевой системой показываются фигурной скобочкой на линии времени:




На таких диаграммах удобно рассказывать истории типа “мы организуем стартап, который создаст САПР, при помощи которого мы затем спроектируем топор, при помощи которого мы потом будем колоть дрова” — и для каждой системы в такой диаграмме понятно, что она проходит довольно долгую жизнь перед тем, как быть использованной.

### Системная медитация

Проведите “системную медитацию”:

> Подумайте о том, чем вы занимаетесь. Но попробуйте подумать об этом культурно, а не "как обычно". В определённых кругах думание не абы как, а определённым способом и по определённому сценарию называют медитацией. Давайте помедитируем над тем, чем вы занимаетесь.
>
> 1. Представьте себе то, что вы создаёте. Если вы не бездельничаете, то вы что-то создаёте. Даже если вы создаёте "сервис", то вы создаете не столько сам этот сервис, сколько то, что потом этот сервис оказывает.
>
> 2. Обзовите то, что вы создаёте, "целевой системой". Обзовите себя и тех, кто трудится над целевой системой вместе с вами (людей, инструменты, помещения) "обеспечивающей системой".
>
> 3. Если вы создаёте много чего разного, то наверняка это либо какой-то повторяющийся цикл создания чего-то однотипного, либо разные части одного большого целого — подумайте, может это большое целое вы тоже создаете регулярно (раз в жизнь — это ведь тоже регулярно, не так ли? Впрочем, и ежедневный цикл, и даже ежечасный вполне возможны). Представьте себе "типовую целевую систему" — воплощающую основные черты того, что вы обычно создаёте.
>
> 4. А теперь ключевое усилие: представьте себе вашу типовую целевую систему от момента ее рождения до исчезновения в небытии. Ничего страшного в представлении системы, находящейся одновременно в разных местах и временах нет:
>
> > а) представьте ее линией времени, проходящей слева направо графиком. Слева зарождение системы, справа её славная или

бесславная кончина. Не держите это представление в голове, нарисуйте перед собой на бумажке.

б) Все изменения состояния системы должны уместиться в этот график, моменты ключевых смен состояний (их легко вычислить не по тому, что делает сама система — она ведь значительную часть времени ничего не делает, например, на стадии замысла или стадии изготовления — а по тому, что меняется то, что делают с системой).

Важно: ваша целевая система — это не точка, кружочек, картинка, прямоугольничек или даже короткий фильм в ваших мозгах (кортексе), а стрелочка со штрихами перед вами на бумажке (экзокортексе). Все состояния, части системы и т.д. — это одна система (так, яйцо, гусеница и бабочка — это одна система, а не три разных). И обозначается одной стрелочкой. Теперь подумайте, и добавьте штришок для отделения стадии пренатального развития яйца. И штришок до оплодотворения. И штришок для сгнивания тушки. Ну, вы поняли уже, в чем фишка.

Не важно: то, что все это переход к 4D-онтологии, и вы мыслите у вашей системы темпоральные части, которые появляются и исчезают (как гусеница в системе бабочки), а система при этом остаётся сама собой. Думайте не в этих сложных терминах, а в терминах стрелочки-жизненного цикла и отрезков между штришками на этой стрелочке — стадиях.

Важный вывод из неважного замечания: система — это не набор фотографий системы в какие-то понравившиеся вам моменты ее времени. Если при слове "яблоко" у вас в голове возникает яблоко, то на бумажке у вас должна быть стрелочка с штришками для переходов от бутончика к цветочку, от цветочка к завязи, от зеленого яблочка к спелому, от спелого — вот тут подумайте сами, тут самое интересное.

5. Подумайте, где вы обычно начинаете принимать участие в судьбе целевой системы (или система начинает принимать участие в вашей судьбе обеспечивающей системы), и где вы эту систему покидаете. Отметьте это на стрелочке.

6. Нарисуйте рядом похожую стрелочку обеспечивающей системы (для "мы", или для "я" — на ваш вкус) — выполните пункт 4 для неё. Можете поупражняться, нарисовать и для этой системы обеспечивающую ее систему (кто делает вас?). Вполне может быть вариант, в котором вы сами себя делаете — ну что же, и так может быть. Или не так. Подумайте над этим.

7. Теперь можно подумать, как называть вашу целевую типовую систему (ту, первую). Собственно, тут волнует не столько название, сколько определение. Определение делается по следующему образцу:

[название типовой системы] — это [название ее родовой системы] [описание специализации]. Рама — это крепление для стекла. Рама — это набор покрашенных деревяшек. Напишите десяток таких определений, которые могли бы дать разные люди, которым система нужна для разных целей.

8. Какое из этих десяти определений сущностное? Какое из этих

определений касается назначения системы? Кому нужна эта система, чтобы ее потреблять (или потреблять оказываемый этой системой сервис)? Сколько этих ролей (система-то у нас типовая, и поэтому я пишу не "людей", а "ролей" — помним про стейкхолдеров!), как они называются? Нарисуйте маленькие фигурки вокруг стрелочки там, где они максимально связаны с системой. Обратите внимание, захотелось ли вам нарисовать эти фигурки как человечков или как стрелочки со штришками. Подумайте над этим вопросом некоторое время.

9. Помните, что вы себя как "обеспечивающую систему" уже нарисовали стрелочкой. Что делает эта ваша стрелочка со стрелочкой системы? Как называется та типовая работа, которую вы делаете с целевой системой? Какая ваша роль? Запишите это на бумажку: [моя роль] [что делаю] [целевая система]. Мама моет раму.

10. Каким методом вы делаете то, что делаете? Осознаёте ли вы, что вы делаете (другая формулировка: смогли ли вы ответить на предыдущий вопрос, назвав при этом несколько альтернативных методов, осознанно вами отброшенных)? Впрочем, это тема отдельной и специальной другой медитации.

Теперь вы знаете, что делаете. Или знаете, что этого не знаете.

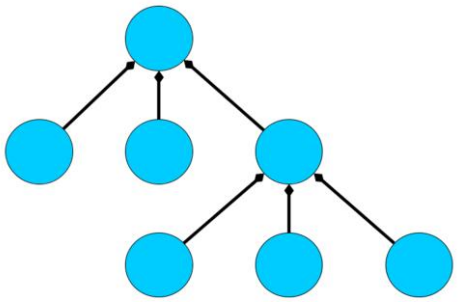
Модифицируйте это упражнение, адаптируйте к вашим потребностям (например, вот одна из недавних модификаций: http://belique.livejournal.com/67802.html

Вот очень сжатый и короткий план более продвинутой системной медитации, но суть её та же самая:

- типовая целевая система как индивид [тест: постучать]
- Использующая система, нужды (и какие будут тесты приёмки)
- Название целевой системы (по основному назначению)
- Типовая целевая система в ходе всего жизненного цикла: нарисовать ЖЦ стрелочкой
- Точки входа-выхода из проекта (ЖЦ проекта)
- Собственная роль в проекте
- Ключевые практики, компетенции в них, доступность технологий

“Сначала как часть надсистемы”

Классическое определение системы говорит о том, что система состоит из взаимодействующих друг с другом частей, и она больше, чем сумма этих частей (свойство эмерджентности). Это классическое определение системы, оно верно, но сбивает с толку. Системные инженеры никогда не начинают рассматривать систему как состоящую из каких-то частей. Нет, системные инженеры понимают, что любая система это холон (целое, состоящее из частей-подсистем, и само являющееся частью целого-надсистемы) — и начинают рассмотрение с того, что “холон это часть другого холона”, а не “холон состоит из частей-холонов”.



В холархии (иерархии холонов, отношения “часть-целое”, показаны на диаграмме линиями с ромбиками на конце “целого”) каждая система сначала характеризуется своей основной функцией в качестве части надсистемы, а уж только затем —из каких она состоит частей, какая у неё конструкция, как она устроена.

Правильно задавать сначала вопрос “в какой надсистеме какую функцию выполняет наша система” и пока нет ответа на этот вопрос — пытаться найти ответ на этот вопрос прежде всего.

Более того, системным инженерам рекомендовано сознательно откладывать понимание внутреннего устройства системы до того момента, как они поймут устройство надсистемы — т.е. пока не разберутся, в каком операционном окружении будет находиться их целевая система, зачем она нужна надсистеме, какая функция целевой системе в её операционном окружении. Это очень трудно, это нужна особая постановка мышления, чтобы вместо того, чтобы говорить “часы состоят из шестерёнок и пружинок” или “часы состоят из чипа, батарейки и дисплея” говорить “часы служат для показа времени на спортивных соревнованиях для 40 тысяч зрителей” или “часы служат для демонстрации высокого социального статуса их владельца, а заодно уж пусть показывают время”. Сначала разбираемся с надсистемой целевой системы, и только потом определяемся с подсистемами — иначе может выясниться, что стейкхолдеры имели ввиду совсем другую систему, у которой будут совсем другие части.

### *Стейкхолдеры. Театральная метафора*

#### Система — это субъективное понятие

В схеме Основ инженерной деятельности “система” представлена прежде всего как “воплощение системы” (system realization — буквально “реализация”, воплощение в реальности). Следуя схеме, мы должны понять, что понятие “система” прежде всего связано с понятием “стейкхолдеров”, которые используют систему в своих интересах. Стейкхолдеры — это те люди (один или несколько с распределёнными в них полномочиями — организации), которые как-то влияют на инженерный проект по созданию, эксплуатации и выводу из эксплуатации системы, а также те люди, интересы которых могут быть затронуты системой.

Стейкхолдеры занимаются какой-то своей деятельностью, и тут в их жизни появляется (или может появиться) система — кому-то эта система даёт новые возможности (например, пользователям, или членам команды — они ведь тоже стейкхолдеры, по определению!), кому-то она мешает (например, конкурентам или борцам за загрязнение окружающей среды). Так вот: “система — в глазах смотрящего”, нет никакого “объективного” способа определить систему без

упоминания стейкхолдеров. Единственный вариант "объективности" — это хорошо организованная субъективность, когда стейкхолдеры договорятся о том, какова их система, что они от неё ожидают.

Любая система определяется (определение системы, system definition — ещё одна альфа из схемы инженерного проекта) так, чтобы это определение было удобно для деятельности стейкхолдера. Какого? В разных случаях разного: поэтому определение системы может существенно отличаться от стейкхолдера к стейхолдеру, речь может идти об абсолютно разных системах и может потребоваться огромная работа по согласованию этих определений. Система для пользователя будет одна, для вора (тоже стейкхолдер!) другая, для распильщика бюджетов третья, для учёного четвертая. Нет никакого способа определить "правильную систему", есть только понимание необходимости специального разбирательства с деятельностями стейкхолдеров и затем предложения определения системы, удовлетворяющего интересам этих стейкхолдеров.

"Говорю система — подразумеваю стейкхолдеров, говорю стейкхолдер — подразумеваю систему", "каждый стейкхолдер имеет деятельность, и в этой деятельности целевая система будет частью надсистемы стейкхолдера" — это проявление схемного мышления, это логика прохождения диаграммы Основ, это самые азы системного подхода.

Конечно, не любые люди, которым система "интересненька", представляют собой стейкхолдеров. Нет, стейкхолдеры — это которые в принципе будут действовать, если им эта система нужна или наоборот, мешает. Обратите внимание, что "наблюдатель" из физики — это не деятель. Зеваки и прочие "наблюдатели" — это не стейкхолдеры. Стейкхолдеры — деятели! Собаки лают, а караван идёт: собаки тут не стейкхолдеры. А вот если купец не оплатит проход каравана, то караван идти не будет. Купец — стейкхолдер.

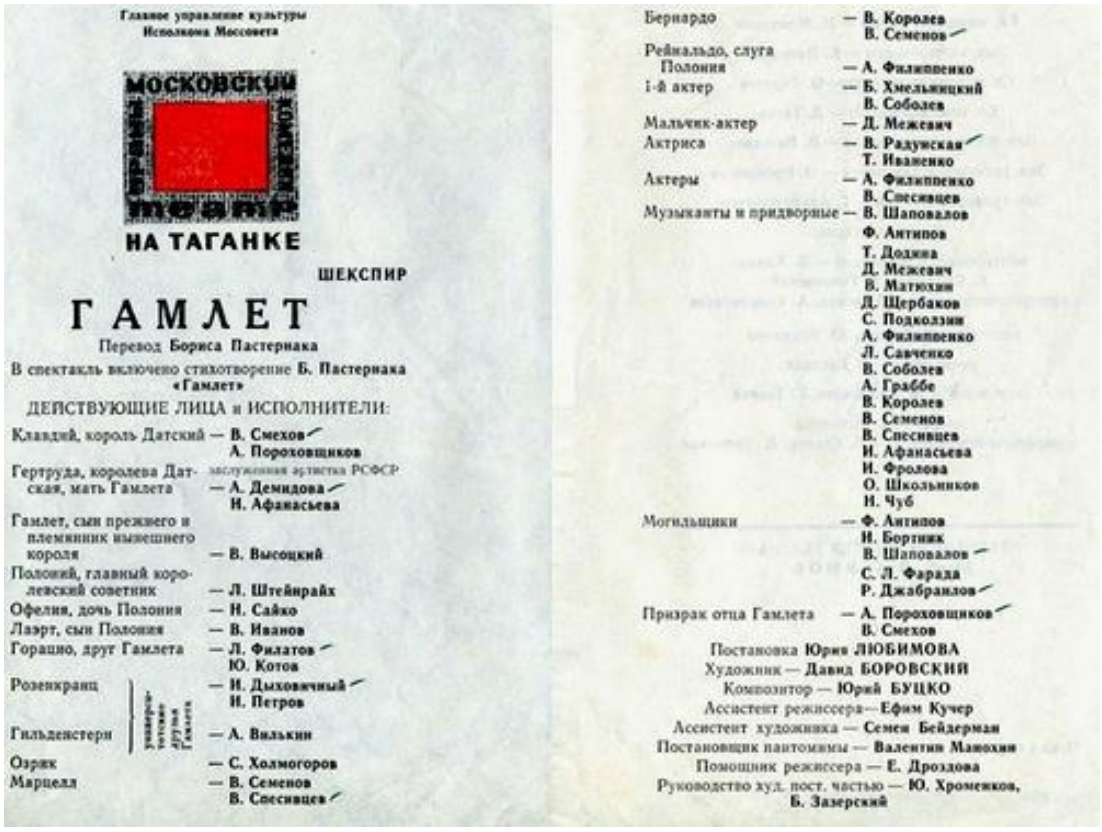
Театральная метафора.

Деятельность — это в чём-то повторяющиеся работы с похожими объектами. Деятельность ведут стейкхолдеры с системами. Одно уникальное действие — это не деятельность. Один конкретный человек — это не стейкхолдер. Деятельность мы описываем как бы "безлично", в культурно-обусловленных типах участвующих объектов, субъектов, действий/операций.

Проще всего обсуждать деятельность как своего рода театральную пьесу, которую разыгрывают по ролям в разных театрах. Несмотря на огромную разницу в интерпретации этих ролей актёрами и их режиссёрами в разных театрах, и даже в одном театре в разные дни, всё-таки есть огромный смысл обсуждать сами пьесы ("методологическую действительность", methodology realm), а не только их отдельные исполнения ("действительность предпринятия", endeavour realm). Театральная метафора сравнивает деятельность с пьесой, практики и даже целый метод — со сценарием этой пьесы. Пьеса играется много раз, деятельность повторяется много раз — хотя каждое исполнение пьесы и каждое действие в чём-то уникальны, но мышление экономится за счёт "выноса за скобки" всего того, что повторяемо. Знание принципов освобождает от знания фактов (тут можно указать на интересную книгу "Программистский камень" http://progstone.narod.ru/reciprocality/r0/index.html — там людей делят на "картостроителей" и "паковщиков" ровно на этом основании: строят ли они карту "принципов", или запоминают каждый отдельный встреченный маршрут, т.е. знают много фактов и их "двадцатилетний опыт работы — это однолетний опыт,

повторённый двадцать раз”).

Программка в театре содержит важнейшую информацию: “действующие лица и исполнители”:



Главное управление культуры Исполкома Моссовета

МОСКОВСКИЙ ТЕАТР ДРАМЫ И КОМЕДИИ НА ТАГАНКЕ

ШЕКСПИР

ГАМЛЕТ

Перевод Бориса Пастернака

В спектакль включено стихотворение Б. Пастернака «Гамлет»

ДЕЙСТВУЮЩИЕ ЛИЦА и ИСПОЛНИТЕЛИ:

Клавдий, король Датский — В. Смехов, А. Пороховщиков

Гертруда, королева Датская, мать Гамлета — заслуженная артистка РСФСР А. Демидова, И. Афанасьева

Гамлет, сын прежнего и племянник нынешнего короля — В. Высоцкий

Полоний, главный королевский советник — Л. Штейнрайх

Офелия, дочь Полония — Н. Сайко

Лаэрт, сын Полония — В. Иванов

Горацио, друг Гамлета — Л. Филатов, Ю. Котов

Розенкранц, Гильденстерн, университетские друзья Гамлета — И. Дыховичный, И. Петров; А. Вилькин

Озрик — С. Холмогоров

Марцелл — В. Семенов, В. Спесивцев

Бернардо — В. Королев, В. Семенов

Рейнальдо, слуга Полония — А. Филиппенко

1-й актер — Б. Хмельницкий, В. Соболев

Мальчик-актер — Д. Межевич

Актриса — В. Радунская, Т. Иваненко

Актеры — А. Филиппенко, В. Спесивцев

Музыканты и придворные — В. Шаповалов, Ф. Антипов, Т. Доджна, Д. Межевич, В. Матюхин, Д. Щербаков, С. Подколзин, А. Филиппенко, Л. Савченко, В. Соболев, А. Граббе, В. Королев, В. Семенов, В. Спесивцев, И. Афанасьева, И. Фролова, О. Школьников, Н. Чуб

Могильщики — Ф. Антипов, И. Бортник, В. Шаповалов, С. Л. Фарада, Р. Джабраилов

Призрак отца Гамлета — А. Пороховщиков, В. Смехов

Постановка Юрия ЛЮБИМОВА

Художник — Давид БОРОВСКИЙ

Композитор — Юрий БУЦКО

Ассистент режиссера — Ефим Кучер

Ассистент художника — Семен Бейдерман

Постановщик пантомимы — Валентин Мамохин

Помощник режиссера — Е. Дроздова

Руководство худ. пост. частью — Ю. Хроменков, Б. Зазерский

Действующие лица — это вдумчивый Принц Гамлет и безумная Офелия. Исполнители — это актёр-стажёр Вася Пупкин в утренних спектаклях и народный артист Василий Петрович Черезколеноногузадерищенский в вечерних спектаклях, плюс Елена Ефимовна во всех спектаклях, и она не болеет и не замещается. Когда говорим о стейкхолдере, то всегда имеем ввиду ту пьесу, которую он играет (ту деятельность, которой стейкхолдер занимается), т.е. стейкхолдеры — это всегда “действующее лицо”, роль.

Конечно, в реальной жизни мы видим только исполнителей — конкретных актёров, а не “роли”. Но обсуждаем мы исключительно роли, если только речь не идёт о качестве исполнения!

Кто говорит фразу “быть или не быть?”. Принц Гамлет, или Вася Пупкин? На момент исполнения роли оба они — один и тот же объект, только называются по-разному и мы обращаем в зависимости от этого внимание на разные свойства этого объекта. Когда речь идёт о “действующем лице”, то мы обращаем внимание на текст и сюжет пьесы, а когда речь идёт об “исполнителе”, то на качество исполнения и доступность исполнителя в момент спектакля.

Мы всегда можем указать Васе Пупкину, что он плохо выучил роль, или играет чужую роль и всяко по-другому дать понять, что “ты не прав, Вася”, если нам известна пьеса, которую он играет. Если пьеса неизвестна, то мы не можем понять — прав, или не прав Вася в своих действиях.

Мы можем даже потребовать заменить актёра-исполнителя (безвестного Пупкина на талантливого народного артиста Черезколеноногузадерищенского), но обычно не можем потребовать заменить действующее лицо (вместо Принца Гамлета вдруг

потребовать вставить в пьесу Бармалея и Бэтмена). Это огромное достижение цивилизации: роли культурно-обусловлены, а исполнители привносят в них личное — и это сливается в одно “исполнение роли”.

Упражнение: пометьте, что неправильно в списке стейкхолдеров (помним, что стейкхолдер — это как один человек, так и группа, так и организация):

1. Системный архитектор

2. Инженер по требованиям, А.К.Щеплынтов

3.Разработчик подсистемы Z, завод “Тепломонтажагрегатика”.

4. Завод “Промпереработка”

5. Геннадий Павлович

6. Пятый цех.

7. Жители района строительства атомной станции.

Подсказка: используйте шаблон “действующее лицо — исполнитель” (при этом действующее лицо должно быть понятно названо для того, кто не знает исполнителя и тех ролей, которые он обычно играет. А исполнитель— про него полезно знать, но он необязателен для многих и многих обсуждений).

Ещё подсказка: стейкхолдеры в проекте должны быть представлены (например, если речь идёт о 10 тысячах потенциальных покупателей продукта, нет возможности работы команды сразу с 10 тысячами человек, но есть возможность работать с представителем этой группы стейкхолдеров — а иногда такого представителя сочиняют по “методу персонажей”, который уже упоминался).

Важно, чтобы все обсуждения проходили в терминах “действующих лиц”, а не исполнителей. Сравните два диалога:

1. “Исполнительское” обсуждение, персоналий:

— Иванов опять чертежи испортил! Он присылает их в формате .dwg и ссылается на Петрова! Сидорова не может работать!

— А что думает об этом “Красшефмонтаж”?

— Его не волнует, лишь бы “Заготбазарбаза” не возражала!

Всё ли вам понятно, если вы случайно попали на совещание? Можно ли задать какие-то уточняющие вопросы по непониманию — и какие? Если вы хорошо знаете всех действующих лиц, то можете ли вы предсказать хоть как-то их предполагаемые реакции в данной ситуации?

2. Стейкхолдерское обсуждение (”действующих лиц”, ролевое)

— конструктор опять чертежи испортил! Он присылает их в формате .dwg и ссылается на расчётчика! Архив не может работать!

— А что думает об этом завод-изготовитель?

— его не волнует, лишь бы поставщик корпусов не возражал!

Стало ли понятней, о чём идёт речь? Какие уточняющие вопросы вы бы задали?

Обсуждение в терминах “действующих лиц” (понимание стейкхолдеров как “деятелей”, а не конкретных личностей-исполнителей) крайне важно для коммуникации: такое обсуждение направляет мысль и позволяет понимать, какие “пьесы” сейчас обсуждаются — какие реплики могли бы следовать, а не только

какие реплики следуют прямо сейчас. Если какой-то “принц Гамлет” вдруг начинает давать реплики “Офелии” — то можно дальше обсуждать: спасает ли он пьесу ввиду неявки Офелии, или просто портит дело.

Одновременное обсуждение ролей и актёров нужно при обсуждении качества их игры, назначении на роли, обсуждении актёрского амплуа (компетенций) и т.д. — это очень важные вопросы, но это больше про командообразование, формирование труппы (leadership — соединение потребных технологией работы рабочих мест с исполнителями работы на этих местах). А когда уже идёт деятельность, то стейкхолдеров лучше называть по их ролям, а не по фамилиям или названиям организаций. Самый тяжёлый случай, это когда люди в проекте знают важность какого-нибудь Василия Петровича (он точно стейкхолдер! Он существенно влияет на проект!), но не могут назвать его функциональную роль в проекте, он поэтому для них “невычислим”, они не знают, что от него ожидать, как реагировать на его действия.

Конечно, если Ельцин у нас долго играл роль президента, то некоторое время после смены играющего роль президента был осмыслен вопрос “А кто у нас сейчас за Ельцина?” — это, конечно, метонимия по отношению “назначен на роль”.

### Позиция

Когда исполнитель застревает в какой-то одной "любимой" роли, и начинает в других ролях действовать так, как он действует в этой роли (т.е. на первом плане оказываются ценности этой роли из соответствующей “пьесы”), то это называется — позиция. Когда исполнитель занимает позицию "инженер", то у него инженерные ценности и когда разрабатывает что-то, и когда воспитывает детей, и когда сидит в парламенте. Когда он в позиции "родитель", то воспитательные ценности и дома среди детей, и в рабочем коллективе, и на шумной вечеринке.

Позиции можно занимать неосознанно (и тогда вами легко манипулировать: любые ваши действия легко вычислимы, ибо действуете уже не вы сами, а какая-то "схема" — позиция и ее ценности). Реакция на указание чьей-то неосознанно занятой позиции разные: "что-то застряла роль в сознании, спасибо, что обратил внимание", или наоборот "какая такая у меня позиция? как так у меня не меняются в разных делах роли? я ведь такой спонтанный!".

А можно занимать позицию осознанно: "сейчас займу вот с такой-то целью такую-то позицию" (выберу себе понятную роль в понятной пьесе, и буду придерживаться ее ценностей в самых разных делах, пока не передумаю). Такой осознанный выбор позиции обычно называется "самоопределением".

Когда исполнитель скачет по разным ролям в одном проекте, как зайчик, то с ним очень трудно наладить коммуникацию: внешний эффект при этом такой, что он непрерывно меняет свой набор ценностей — что было для него ценным в его предыдущей позиции пять минут назад вдруг перестаёт быть значимым, но зато появляются какие-то новые претензии. Это можно назвать "какой гибкий человек, никто его подловить не может", а можно и "какой скользкий".

Многие люди не скользкие, потому как застревание в их позиции происходит вне их сознания, просто из-за их застревания в системе ценностей того дела, которым они подолгу занимаются — поэтому они выглядят как принципиальные люди, отстаивающие какие-то свои принципы. Если у них своего дела нет, то они могут так же бессознательно "не держать позицию", и выглядеть поэтому скользкими и беспринципными: они никогда не “принцы Гамлеты”, они всегда Васи Пупкины, с

ними невозможно играть пьесы, с ними трудно работать в проекте с разделением обязанностей.

Люди, которые осознают свои застревания в (профессиональных, социальных, семейных) ролях могут выбирать — занимать ли им какие-либо позиции, или менять их в зависимости от ситуации. Люди, которые осознают чужие застревания в ролях, часто могут понять мотивы тех или иных действий и высказываний стейкхолдеров [тут я использовал метонимию: правильно, но длинно было бы говорить об "исполнителе роли стейкхолдера" — рабочем продукте "исполнитель" для альфы "стейкхолдер". Но об альфах говорят обычно с использованием метонимии, считая их реальными объектами: альфа и рабочий продукт сливаются в один объект].

В большом числе случаев "позиция" определяется профессией — и названия распространённых "ролей" в деятельности это очень часто названия профессий (профессиональных дисциплин): менеджер, инженер по требованиям, эккаунт-менеджер (занимающийся стейкхолдерами проекта и возможностями — клиент-менеджер).

В инженерных проектах необходимо всегда понимать позицию всех исполнителей — позиция исполнителя роли стейкхолдера может как соответствовать требуемой от данного стейкхолдера, так и не соответствовать ей. Поэтому на всех совещаниях и при прочтении всех документов проекта нужно понимать: какой это стейкхолдер проекта, какой исполнитель стейкхолдерской роли, и какую позицию занимает этот исполнитель (если он её, конечно, занимает). Это понимание должно быть абсолютно осознанным и его желательно документировать (затруднения с документированием часто показывают недостаточную продуманность вопроса — "собака всё понимает, но сказать не может", рука зависает над клавиатурой, но не пишет!).

### Работа со стейкхолдерами

Хороший анализ видов внешних стейхколдеров при крупных продажах (например, инженерного оборудования — в отличие от розничной продажи игрушечной машинки) дан в книге Нила Рэкхема "Стратегия работы с клиентами в больших продажах" — http://qame.ru/book/sale/strategiya_raboti_s_klientami_v_bolshih_prodazhah/%d0%a1%d1%82%d1%80%d0%b0%d1%82%d0%b5%d0%b3%d0%b8%d1%8f%20%d1%80%d0%b0%d0%b1%d0%be%d1%82%d1%8b%20%d1%81%20%d0%ba%d0%bb%d0%b8%d0%b5%d0%bd%d1%82%d0%b0%d0%bc%d0%b8%20%d0%b2%20%d0%b1%d0%be%d0%bb%d1%8c%d1%88%d0%b8%d1%85%20%d0%bf%d1%80%d0%be%d0%b4%d0%b0%d0%b6%d0%b0%d1%85.djvu (читать этот файл нужно программой-читалкой файлов DjVu, например, WinDjView — http://www.koob.ru/about/windjview-setup.exe). В этой книге говорится, что в крупной организации за простым словом "клиент" могут скрываться самые разные стейкхолдеры — и со всеми ними нужно работать по-разному. Так что "нашими клиентами являются поликлиники" говорить можно только в самых общих стартапных презентациях. В реальной жизни внутри этой поликлиники обнаруживается много разных стейкхолдеров — и главный врач, и старшая сестра, и генеральный директор, и главный айтишник, и лаборанты, и пациенты, и невидимый обычно инвестор-владелец. Когда вы говорили "нашими клиентами являются поликлиники", то кого из них вы имели ввиду? Для каждого из них нужно уметь отвечать на разные вопросы, подавать материал на разном уровне детальности, хвалить систему за разное, по-разному отстраиваться от конкурентов,

вести переговоры на разных стадиях продажи.

Там же проводится идея, что стейкхолдеров нужно проводить через разные их состояния: стейкхолдеры не только опрашиваются, с ними работается как и с любой другой альфой. Состояния, через которых проводится стейкхолдер-покупатель в методике Нила Рэкхема очень похожи не те состояния, через которые стейкхолдеров проводят при так называемом «снятии сопротивления» в теории ограничений (ToC) Голдратта (пример -- http://www.tocpractice.com/ru/wiki/kak-sdelat-prodavtsa-prodayushchim-ili-pervyi-opyt-pri-razrabotke-i-vnedrenii-programmy-po-upra). И тут нужно обратить внимание, что практики проведения стейкхолдеров через последовательности состояний очень похожи и у Нила Рэкхема, и у Голдратта. Более того, они похожи для случая крупных продаж и организационных реформ (где нужно «продать» реформу отдельным позиционерам).

### Граница системы и деятельностная субъективность её проведения

Любая система имеет свою "границу" — так говорят о том, что какие-то объекты мира включаются как части в состав "целого" системы, а какие-то объекты не включаются. Эта "граница" как раз и разграничивает все объекты мира (включая как системы в операционном окружении, так и просто любые другие объекты окружающего мира, даже не взаимодействующие с системой) и объекты мира, входящие в целевую систему. Говоря о "системе X" тем самым мы подразумеваем существование какого-то куска мира, называемого X и всего остального мира за пределами этого куска. "Система" — это такой способ указать на специальным образом выделенный кусок мира в его противопоставлению тому миру, который остался за границей системы.

При определении границы системы нужно понимать, что система задаётся субъективно — само выделение системы из окружающего мира должно быть таким, чтобы удобно было рассуждать о системе и проводить с ней какие-то действия в рамках деятельности стейкхолдера (и лучше бы сразу договариваться о границах системы так, чтобы было удобно сразу нескольким стейкхолдерам).

Система определяется стейкхолдерами субъективно — это означает "деятельностную субъективность" (помним, что деятельность повторяема — у неё есть "сценарий", речь не идёт о единичном действии), пристрастность обусловленной сценарием роли, пристрастность "действующего лица" — ролевая пристрастность, окрашенная талантливостью или бесталанностью исполнителя-актёра. Деятельность системного инженера как стейкхолдера отличается от деятельности менеджера по продажам инжиниринговой компании, и обе они отличаются от деятельности операционного менеджера. Система для целей этих деятельностей будет определена "субъективно", именно с точки зрения этих стейкхолдеров. Но это "деятельностная субъективность", её вполне можно объективизировать, если известно, какая именно пьеса играется, какая деятельность выполняется, что можно ожидать от стейкхолдеров в соответствии с их деятельностными ролями и позициями. Поэтому граница системы определяется субъективно как удобная для целей деятельности стейкхолдеров, деятельностно-субъективно, а не личностно-субъективно. Объективация же границы проходит как согласование определения системы между различными стейкхолдерами: стейкхолдеры подстраивают свои деятельности так, чтобы в них появлялась система в одинаковых для всех границах. Но не нужно ожидать, что граница системы для всех стейкхолдеров одинакова в момент начала проекта. Скорее, верно обратное: хотя все стекхолдеры и будут называть систему (какую-нибудь "атомную

электростанцию” или “прибор анализа крови”) одинаково, представлять себе они её будут по-разному, и по-разному проводить границу между системой и её средой (системами в операционном окружении).

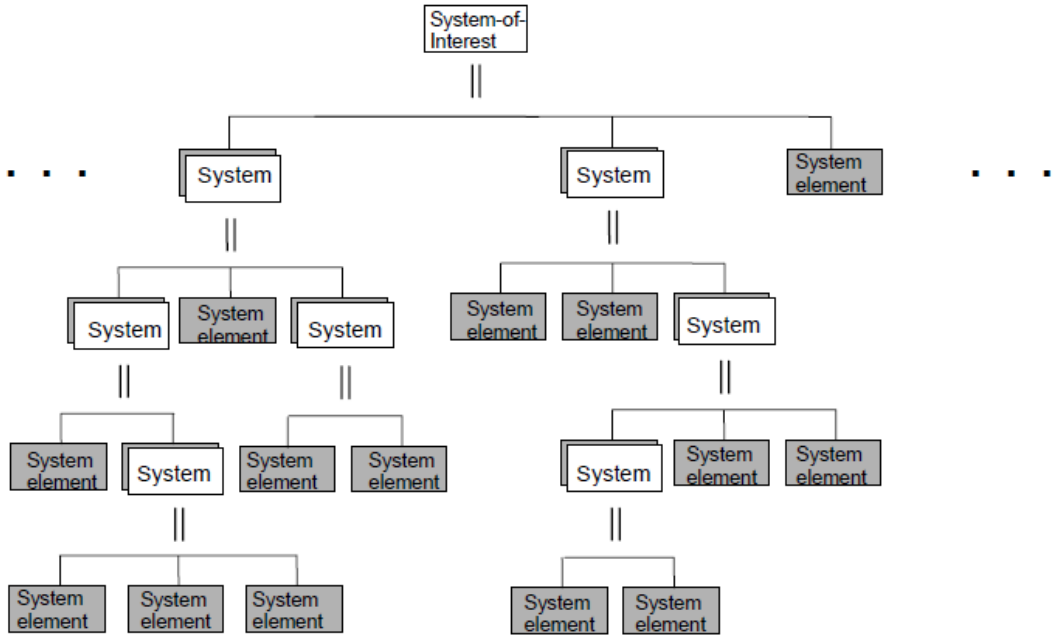
Личная субъективность актёра Пупкина или даже народного артиста Черезколеноногузадерищенского тут мало кого интересует (хотя иногда интересно сообразить, что это за пьесу они играют и какую в этой пьесе роль — критика тогда их “личной субъективности” будет проще). В деятельности нас интересуют мнения стейкхолдеров, когда они “в образе” и играют честно, а не когда они актёры-без-роли (хотя, конечно, есть множество ситуаций, когда обсуждается сама деятельность и распределение ролей в ней — но это мы пока не рассматриваем).

Какова граница солнечной системы? Эта граница проходит по “диску” планетарных орбит, или же сферична, игнорируя планеты? Правильный ответ на этот вопрос — это сначала задать уточняющий вопрос: какой стейкхолдер определяет границы, и какова деятельность этого стейкхолдера, чтобы для него было осмысленно проводить границы именно таким способом — каким образом целевая система важна для его деятельности? “Изучить систему” при этом — не ответ, ибо что в системе нужно изучить? Цвет солнечной системы? Пригодность для жизни? Наличие ангелов? Пригодность для стихосложения? “Дать краткое описание солнечной системы” — тоже не ответ, ибо для кого эти “краткие описания” будут полезны, как и кто их будет использовать? Но уже для деятельности по организации радиосвязи с каким-нибудь космическим зондом вопрос о границах солнечной системы может быть осмысленным. И для людей, задающихся вопросом о потенциальной космической экспансии. Вообще, определение границ солнечной системы мало кого интересовало, пока не появился деятельностный повод: пиар-компания по поводу выхода Voyager 1 за пределы границ солнечной системы в межзвёздную среду. Тогда были уточнены представления разных стейкхолдеров, заинтересованных в солнечной системе как целом и которые на неё хоть как-то “влияют” (например, описывая её для различных целей) — http://www.astronet.ru/db/msg/1171222, http://habrahabr.ru/post/193562/

Итак — “система в глазах смотрящего”, определяется субъективно. Эта субъективность деятельностная (то есть повторяемая, но повторение не делает “субъективность” “объективностью”). Любая “объективизация” — это результат договорённостей разных стейкхолдеров, согласовывающих свои деятельности и картины мира.

### *“Просто” системы и системы систем.*

Различают “просто” системы (system) и “системы систем” (system of systems, SoS). Оба варианта с точки зрения самого воплощения системы как физического объекта в реальности (system realization) представляют собой какие-то холархии (иерархии по отношениям “часть-целое”, разбиения/breakdowns). В обоих случаях “подсистемы” очень часто называются точно так же: “системы” (и поэтому новички в системной инженерии часто пытаются обозвать просто систему “системой систем” — но это ошибочно). Вот, например, диаграмма “просто системы” из ISO 15288 — обратите внимание, что термины “подсистема” и “надсистема” не используются, чтобы подчеркнуть единообразность понимания “системы” на всех уровнях разбиения системы на части-системы и части-элементы (в ISO 15288 элементом называется та часть системы, которая будет оставаться в данном инженерном проекте “чёрным ящиком” и поэтому дальше не будет разбита на части — например, закуплена целиком или изготовлена как целое из какого-то материала):



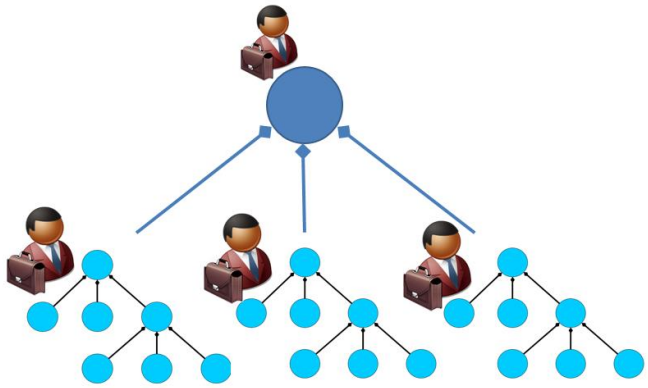

Этот рисунок структуры системы говорит, что ISO 15288 рассматривает целевую систему как набор из частей-систем и частей-элементов и продолжает разбиение систем на части-системы и части элементы. Но неправильно такие картинки называть “система систем” (system of systems, SoS), ибо этот термин закреплён за другой ситуацией. Системой систем называют такую систему, которая (критерии Maier):

- Имеет независимое управление её систем-элементов (нет, кому скомандовать общее развитие-модернизацию)
- Независимая работа элементов (нет, кому скомандовать работу в общем сервисе)
- Эмерджентность от объединения в систему (кто-то желает получить от целевой системы систем функцию, которую невозможно получить от работы с отдельными входящими в систему систем элементами, и требуется совместная работа этих элементов).
- Эволюционное развитие (понимание того, что будет происходить в системе систем на каждом следующем шаге проекта требует исследований, ибо нет точки, которая знает as built для всех)
- Географическое распределение элементов

Эти критерии различаются, конечно, в разных инженерных школах, но общее остаётся: обычные “системы” подразумевают централизованное “владение” системой — наличие стейкхолдеров, полномочных принимать решения по всем частям системы, полномочных распоряжаться всем, что в границах их системы. Это традиционный случай: автомобиль с двигателем и колёсами, железнодорожный мост и компьютер — это типичные “просто системы”, у них есть свои системные инженеры, которые полностью определяют их функции, конструкцию, интерфейсы с системами в операционном окружении, планы по модернизации и выводу из эксплуатации. У каждой из этих систем есть один хозяин, один владелец.

А вот в системе систем каждая из систем имеет своего хозяина, и система может функционировать автономно, без вхождения в систему систем. Тем самым разница

между “просто системой” и “системой систем” определяется не через особую структуру или конструкцию системы, а через наличие независимых друг от друга стейкхолдеров, определяющих и создающих системы, а затем независимо использующих их.



В системе систем важны прежде всего владеющие частями-системами люди-стейкхолдеры, именно они делают систему систем особым случаем.

В армии NATO сейчас больше говорят не о системной инженерии, а о системо-системной инженерии (system of systems engineering), потому что вся армия должна действовать в бою как единое целое — но это оказалось крайне сложно обеспечить: каждый род войск имел своё независимое финансирование много лет, свои планы развития, свои типы вооружений. В итоге флот, авиация, пехота, космические войска получили несовместимое оборудование и вооружение — и никакими силами нельзя было создать из этих несовместимых между собой систем-элементов систему систем, ведущую бой как единое целое.

NATO выделило четыре типа систем систем, отличающихся степенью их автономности:

- управляемые (directed), в которых есть назначенный архитектор, который может выдавать приказы составляющим системам и распоряжается ресурсами.
- подтвержденные (acknowledged), в которых признаваемый архитектор есть, но он может только уговаривать составляющие системы самоизмениться согласно разработанной им архитектуре.
- сотрудничающие (collaborative), в которых все системы договариваются друг с другом по каждому чиху, но архитектора, менеджера проекта или аналогичного выделенного органа управления нет.
- виртуальные (virtual), в которых системы вообще не знают друг о друге ничего и не влияют друг на друга явно.

Был выведен основной способ работы с системами систем: совместная постепенная асинхронная эволюция (модернизация) входящих в систему систем автономных систем — ибо согласованность и синхронность изменений в этих автономных системах крайне сложно обеспечить: даты утверждения проектов модернизации будут различаться, получаемое на модернизацию финансирование будет выделяться в разные моменты и нельзя будет гарантировать его достаточность, системные инженеры могут иметь разные мнения по поводу взаимодействия их систем с другими системами в составе системы, хозяева систем могут сопротивляться переменам (ибо их вполне может удовлетворять и автономная

работа их систем в их надсистемах, а желание какого-то стейкхолдера системы систем об объединении автономных систем в общую систему систем они могут не разделять).

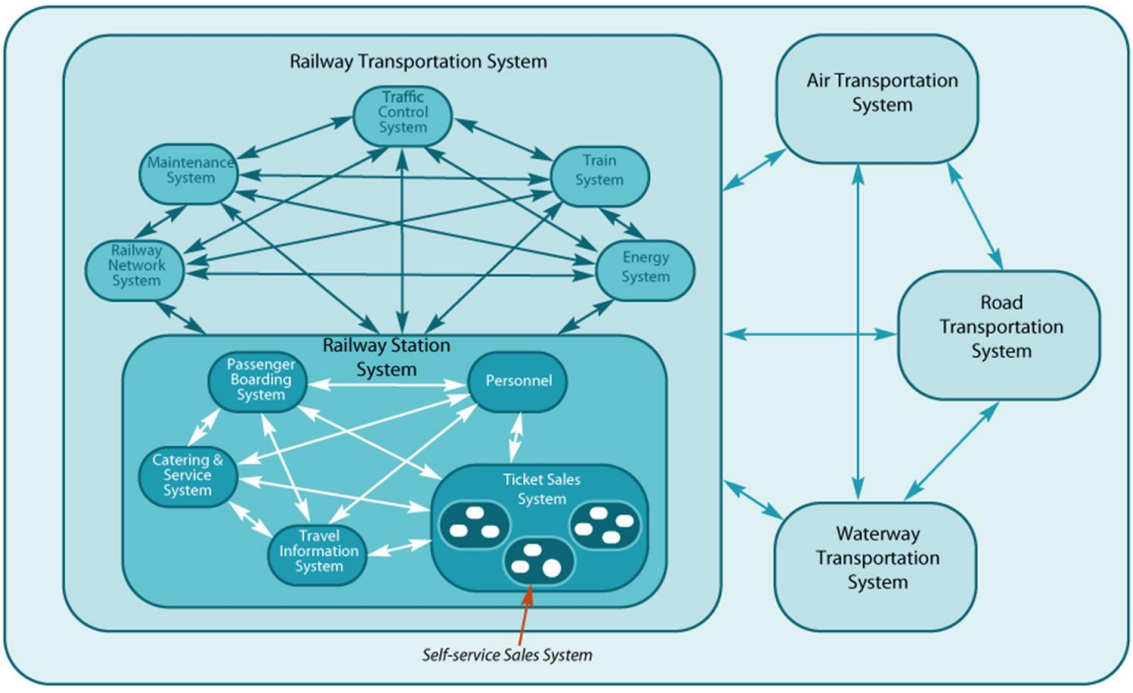
В системо-системной инженерии нет никаких чудес: в ней по факту нет никаких своих понятий. Поскольку работа со стейкхолдерами является главной, то к самой обычной системной инженерии в больших количествах добавляются заимствования из гуманитарных дисциплин. Работы по системо-системной инженерии объединяют с использованием системного мышления достижения отдельных гуманитарных дисциплин: социологии, политологии, психологии, менеджмента, конфликтологии.

Тем не менее нельзя системо-системную инженерию считать "системным менеджментом", ибо в ней таки ставятся задачи не столько главным образом по созданию системы из людей, сколько задачи по созданию главным образом технических систем, но с активным участием людей — что требует изменения инженерных практик и методов.

Навигация по уровням холархии "zoom — select".

Понимание того, что любая система входит в холархию позволяет системному инженеру хорошо ориентироваться в сложном мире: ни на секунду он не теряет контекста, оставаясь способным обсуждать как самый маленький винтик в самом маленьком приборе, так и совсем огромные системы планетарного масштаба. От этих "скачков масштаба" он не сходит с ума, для него это самая обычная процедура: поменять целевую систему в ходе обсуждения на надсистему или подсистему — главное, чтобы он это делал осознанно. Системный подход даёт нам не только оператор "select" (выбора объекта для действия), но и способ для зума — как в фотоаппаратах, можно выбрать подходящий масштаб разбирательства с ситуацией.

Harold "Bud" Lawson приводит следующий пример для транспортной системы:




В транспортной системе мы сначала можем обсуждать мультимодальные перевозки и конкуренцию независимых друг от друга транспортных систем (так, трубопроводный транспорт конкурирует в перевозке нефти с железнодорожным транспортом — для их владельцев они враги в операционном окружении, для желающего перевезти нефть из одной точки мира в другую они части одной

мультимодальной транспортной системы). Это планетарные масштабы, или масштабы целой страны.

В одной из подсистем транспортной системы как системы систем можно обсуждать железнодорожную систему — все эти поезда, энергетику железной дороги, управление движением поездов и т.д. Это всё еще во многом системы систем: владельцы рельсов и владельцы подвижного состава имеют зачастую разное финансирование, разные цели, разные планы развития, хотя они и понимают, что им бы неплохо быть одной системой — ибо рельсы без поездов бесполезны, равно как и поезда без рельсов, поэтому им нужно как-то договариваться. Если взять одну из подсистем железной дороги — систему железнодорожной станции, то она уже практически классическая инженерная система, хотя при ближайшем рассмотрении это может оказаться не так: огромное количество оборудования просто будет размещено на территории железнодорожной станции, и будет развиваться независимо друг от друга. Так, подсистема продажи железнодорожных билетов вполне может быть независимой от конкретной железнодорожной станции, хотя и размещаться на территории этой станции. Часть этой подсистемы — это подсистема автоматов по продаже билетов. Эти автоматы тоже каждый могут быть рассмотрены как отдельные системы. Эти автоматы далеки уже от "системы систем" и уже представляют собой классические киберфизические системы, у разных частей этих автоматов уже нет автономного функционирования и автономного ими владения и развития. Винтики в корпусе такого автомата — это тоже системы, обычные физические системы. Вот так, в одном абзаце и одной маленькой картинке мы проходим рассмотрение ситуации от планетарных или страновых масштабов до маленького винтика, и при этом не сходим с ума, не теряем нити рассуждений, чётко понимаем каждый раз предмет обсуждения и масштабы проблем. Навигация (перемещение в рассмотрении) по уровням холархии чрезвычайно мощный инструмент системноинженерного мышления.

### Системы с участием людей: осторожно!

Достижения системной инженерии как технической дисциплины очень хорошо работают с физическими системами. С киберфизическими системами системная инженерия уже работает не так уверенно: сейчас стоит задача по факту объединения системной инженерии, инженерии управляющих систем (control engineering) и программной инженерии (software enginering) — и есть различные мнения по тому, как это лучше сделать (см. подробности в тренде "роботизация всего" — http://ailev.livejournal.com/1098827.html). Но если ISO 15288 говорил, что "в состав системы люди могут как входить, так могут и не входить", то на сегодняшний день очевидно, что какие-то более-менее сложные системы без людей рассматривать нельзя. А поскольку каждый человек владеет как минимум сам собой как подсистемой, то рассмотрение "системы систем" вместо "просто системы" возникает сегодня много чаще. Системы сегодня рассматривают не столько уже как киберфизические, но как социотехнические или кибер-физико-человеческие (cyber-physic-human).

Если пустить автомобиль по льду, то можно довольно легко понять, как он будет двигаться. Если к автомобилю приложить компьютер, управляющий тормозами и двигателем, то уже не так легко предсказать, как будет двигаться автомобиль по льду. А если к компьютеризированному автомобилю приложить ещё и человека с его собственными целями, то движение автомобиля станет вообще не очень предсказуемым.

Ступень ракеты вдруг не может решить по ходу дела, что она летит не на Марс, а на Луну. А вот человек, как компонента системы, вдруг может проявить такую самостоятельность и начать действовать совсем не так, как ожидают от него системные инженеры, более того, он может сознательно противодействовать тому, как из него делают "винтик" надсистемы, и не факт, что он будет в этом противодействии глупей системных инженеров (если ему будет очень нужно, он и ночью лишний раз о своём противодействии подумает, и друзей пригласит обсудить ситуацию и способы уклониться от предписаний системного инженера). Поэтому для инженерной работы с учётом интересов стейкхолдеров (людей), владеющими автономными подсистемами и переходят к системо-системной инженерии.

Одной из поддисциплин системной инженерии также является организационная инженерия, которой будет посвящена отдельная тема.

## 6. Воплощение системы: компоненты, модули, размещения

Альфа воплощения системы — это сама система, состоящая из вещества и полей. Воплощение системы входит (в том числе потенциально входит, если речь идёт о какой-то серийной продукции) в какую-то использующую надсистему, а также состоит из частей.

При работе с воплощением системы от разных стейкхолдеров можно (а точнее — нужно!) получить разные ответы на два основных вопроса "что это за система" (игнорируем части системы: система как "чёрный ящик"), и "из чего состоит система" (интересуемся частями системы: система как "прозрачный ящик"), это нужно отчётливо осознавать.

Конечно, воплощение системы связано с определением системы (минимальное определение системы — это имена, которыми мы определяем части системы). По большому счёту, мы будем сейчас рассказывать о структуре системы (не молча же размахивать руками и тыкать пальцами в разные части системы! Придётся делать описание воплощения системы!) — но целью нашей пока является не столько задание описаний ("как говорят о системе"), сколько выяснение онтологической природы воплощения системы — "что такое сама система, из каких частей она состоит". Так что про собственно определение системы ("как определять систему", а не "из чего состоит система") разговор будет позже, в следующем разделе.

### *Многерица*

Разные люди видят в одной и той же системе абсолютно разные объекты — ибо они смотрят на них согласно своей роли, своей позиции. Это суть системного подхода, это и есть холизм (полноты рассмотрения с разных позиций стейкхолдеров по отношению к деятельности) против редукционизма (когда всё-всё связанное с системой можно объяснить с позиции одного научного предмета).

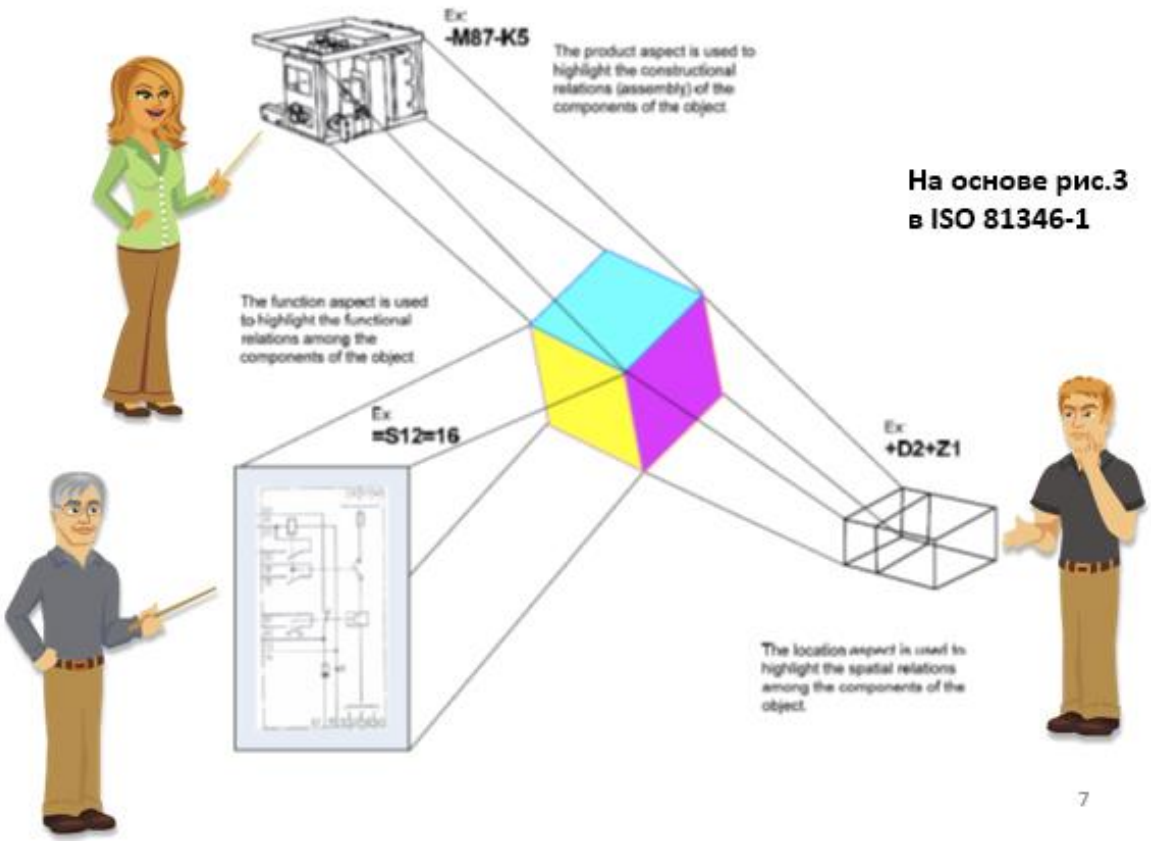
В христианстве есть довольно сложное понятие троицы: бог представляется одновременно и единым, и существующим в трёх ипостасях (отца, сына и святого духа). Проблема в том, что думать о боге нужно одновременно и о как едином, и как об отдельных ипостасях.

То же самое есть и во многих других религиях, хотя там необязательно троица. Так, в даосизме единое (дао) одновременно представляется как двоица (противоположности инь и янь). Впрочем, некоторые и тут видят троицу:


“Он принял свой аспект и поднял атрибут” ("He has taken on his Aspect and raised up an Attribute” — ”Бог/князь/Lord света”, Роджер Желязны — это уже фантазийные перепевы индуизма и буддизма). Все религии как-то учитывают возможность различных обликов/аспектов/ипостасей для какой-то божественной сущности — и тут нужно добавить, что религиозность и божественность тут не при чём. Но метанойя думать об одном как о разном и как об одном и том же вам потребуется. Трудность мышления в том, что нужно о какой-то системе одновременно думать как о чём-то, совмещающем разные свои ипостаси, так и как об отдельных ипостасях — в том числе при той трудности, что ипостаси этой системы нередко имеют ещё и разные имена, обозначающие систему-как-ипостась (хотя одинаковые имена для разных ипостасей встречаются даже чаще).

Мы уже знакомились с ситуацией, когда Принц Гамлет, народный артист и Черезколеноногузадерищенский могут быть одним человеком, но называться по-разному в зависимости от того, что нам от него нужно — понять, какая фраза будет следующая в его пьесе, когда он планирует выучить новую роль в новом спектакле или есть ли у него дети. То же можно сказать и о системе: “измеритель давления”, “манометр KLM-23 завода “Давижмимонтажавтоматика”” и “датчик в пятом ящике на третьей полке склада номер 4” вполне могут оказаться одним и тем же прибором — но разные имена свидетельствуют о том, что мы планируем совершенно разные действия с этим прибором, поэтому для нас один и тот же прибор выступает в разных ипостасях и имеет поэтому разные имена.

Для разных стейкхолдеров система будет представляться в своих аспектах-ипостасях совершенно по-разному — но при этом оставаться целостной, холистичной, целокупностью всех своих ипостасей/аспектов.




Одному стейкхолдеру на этой картинке нужно знать, как работает эта система, и он выбирает для этого одни объекты-части. Другому стейкхолдеру нужно систему собирать, третьему — какое место она будет занимать. И поэтому каждый стейкхолдер по-своему находит части в так “односторонне” (со стороны нужд его деятельности) понимаемой системе.

Это “не единое” просто нормально, так и должно быть! Единым может быть только единонемыслие (Салтыков-Щедрин). Много разных профессионалов по-разному определяют одну и ту же систему, это нормально.

Кстати, шутка про наполовину полный стакан существует и для системных инженеров: “стакан вдвое больше, чем нужно” (помним, что системный инженер переделывает систему — все инженерные описания нужны именно для этого!).

### Сколько разных ипостасей в одной системе?

В одной системе огромное количество разных испостасей: разных деятельностей у людей много, каждая из этих деятельностей потенциально по-своему выделяет части в системе, по-своему называет систему, по-своему употребляет систему. Тем не менее, какие-то общие типы основных ипостасей системы существуют, хотя и по-разному определяются в разных школах мысли:

- ISO 15926 – две основных: функциональные объекты, физические объекты. Остальные могут вводиться по потребности.
- IEC 81346 – «по меньшей мере» три (функция, продукт, место)
- Книга по документированию архитектуры Garlan et. al (http://libgen.org/get?nametype=orig&md5=abb4676b7b1ddfa74e33b7799ebbb61a) – три стиля (компоненты, модули, размещения, и разные варианты внутри каждого стиля)
- СМД-методология – «по меньшей мере» пять (процессы, элементы и связи, внешние функции, морфология, материал) — http://webcache.googleusercontent.com/search?q=cache:CLJYHzTDPj0J:www.mmk-documentum.ru/glossary/6
- … и так далее, в среднем 3-7 ипостаси (впрочем, “ипостаси” тоже называются везде по-разному) в разных школах системного мышления.

В нашем курсе мы примем, что системы имеют минимально три ипостаси/аспекта — компонент, модулей и размещений. И запомним, что этих ипостасей/аспектов всегда больше, плюс сами эти ипостаси/аспектов в чистом виде не встречаются и обычно тесно сплетены друг с другом (”гибридны”).

### Принцип разделения интересов

То, что систему нужно рассматривать много раз для разных целей, сплетая каким-то образом эти разные рассмотрения, называется в культуре очень по-разному:

Принцип разделения интересов (separation of concerns) — в программировании Edsger W.Dijkstra в 1974 году сказал: “Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects. We know that a program must be correct and we can study it from that viewpoint only; we also know that it should be efficient and we can study its efficiency on another day, so to speak. In another mood we may ask ourselves whether, and if so: why, the program is desirable. But nothing is gained —on the contrary!— by tackling these various aspects simultaneously. It is what I sometimes have called "the separation of concerns", which, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts, that I know of. This is what I mean by "focusing one's attention upon some aspect": it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect's point of view, the other is irrelevant. It is being one- and

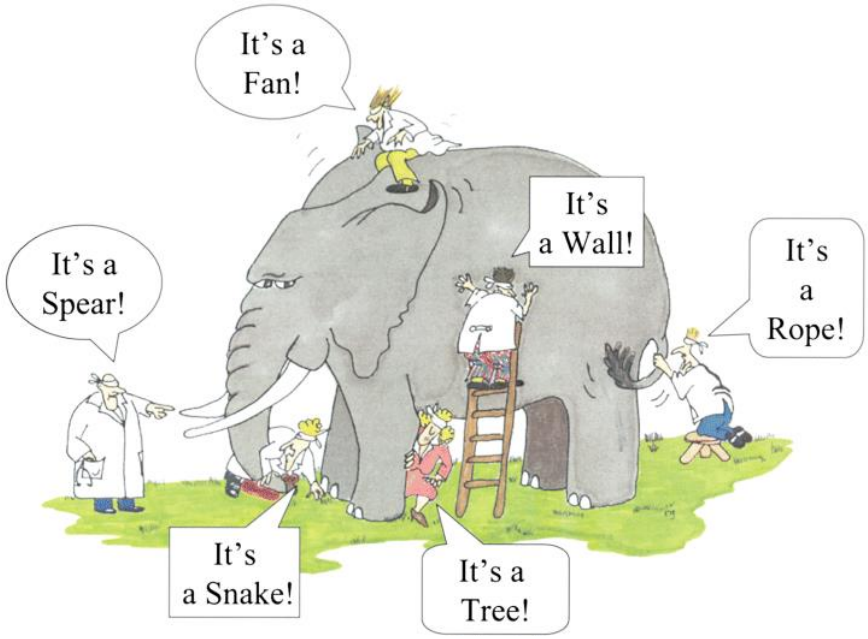
multiple-track minded simultaneously".

Обратите внимание на слова: "aspect" (аспект), "point of veiw" — они неминуемо появляются во всевозможных вариантах описания этой техники. Мы будем переводить view как "группа описаний" (а внутри группы описаний у нас будут различные models/модели — это мы следуем терминологии ISO 42010).

Конечно, это не один человек последовательно (часто — переходя в разные роли) может интересоваться разными аспектами системы, но разные люди — исходя из тех стейкхолдерских ролей, в исполнении которых они профессионализируются.

Что же происходит со всеми этими разными пониманиями, разными описаниями, разными аспектами в момент представления целостной системы? В этот момент иногда говорят о том, что эти аспекты, описания, представления "соответствуют друг другу" (correspondence — так в стандарте ISO 42010), иногда говорят о том, что они переплетаются (weaving — так в аспект-ориентированном программировании).

В любом случае: как съесть слона? По кусочку за раз. Как описать систему? По одной группе описаний/аспекту/ипостаси за раз — не теряя из виду, что все эти описания соответствуют одному целому:




Закрытый и открытый миры

Есть два допущения по поводу описания мира:

- Закрытый мир (close world). Так, например, описывают мир в предметной области баз данных: мир таков, каким он описан в базе данных (проектной документации). Если написано, что системы бывают чёрные и красные, значит никаких зелёных систем не бывает.
- Открытый мир (open world): если чего-то не сказано, что-то не описано, то это вполне может быть дописано потом. Если написано, что системы бывают чёрные и красные, то неизвестно, бывают ли зелёные системы. Вполне возможно, что бывают, просто о них ещё не говорили.

В настоящее время медленно-медленно происходит переход от описаний закрытого мира к описаниям открытого мира. В программировании, в частности, это происходит путём сдвига от "базоданческих" только описаний данных к

семантическим и онтологическим описаниям (http://en.wikipedia.org/wiki/Open_world_assumption — и обратите там внимание, что наш "стейкхолдер" заменён философским словом агент/agent — http://en.wikipedia.org/wiki/Agency_%28philosophy%29. Это не "наблюдатель", а "деятель", хотя предметом обсуждения и является тут "рассмотрение", "наблюдение", "описание".

Два типа "целого"

Про то, как описывать систему, много подробней будет в следующем разделе. Сейчас нас волнует только то, как описывается воплощение системы как целое — но с учётом того факта, что разные люди видят даже это "целое" как ипостаси!

И тут нужно не путать два использования слово "целое" (единое, совокупное, целокупное и т.д.):

- По отношениям часть-целое (разбиения) — это когда система представляется состоящей из частей и сама является частью другой системы. Холархии, "зум" и "уровни рассмотрения" как уровни разбиения системы на части, эмерджентность от взаимодействия частей. Когда говорят о частях и целом, обычно имеют в виду воплощение системы.
- Целое, проявляющее себя в разных ипостасях, рассматриваемое разными стейкхолдерами по-разному — в системном подходе это "мультидисциплинарность", объединение описаний, отсутствие редукционизма (сведения к одной дисциплине), холизм (та же "целостность", но не про части! Про "разносторонность"!). Когда говорят о частных и целостных описаниях, то обычно имеют в виду определение системы (разные описания системы).

Не путайте эти две разных "целостности" системы.

Конечно, сам разговор о воплощении системы подразумевает какое-то описание этого воплощения — как минимум, описание самого воплощения: называние его по имени, учитывая какой-то аспект/ипостась. Именно поэтому, говоря в этом разделе про воплощение системы мы вынужденно касаемся и определения системы, её частей-подсистем и элементов, надсистемы, использующей системы, систем в операционном окружении, обеспечивающей системы и т.д. — как минимум, называние их с учётом аспекта/ипостаси.

### *Компоненты, модули, размещения*

Системный инженер должен один раз рассмотреть систему как целую со всеми ипостасями — но для этого он должен один раз рассмотреть систему как компоненту (и состоящую из компонент), один раз как модуль (состоящий из других модулей), один раз как размещение (состоящее из других размещений) — и так далее в соответствии с принципом разделения интересов, но ни на секунду не забывая при этом целокупность системы во всех её ипостасях.

Тут нужно понимать, что человеческий разум несовершенен, и точно выделить именно компоненты, модули, размещения и т.д. мало кому удаётся: в реальной жизни всё это путается, но всегда можно разобраться, если помнить про 4D экстенсионализм и доводить все рассуждения до разбирательства с воплощением системы, т.е. к имеющим пространственно-временную протяжённость (extent) реальным физическим объектам из вещества и полей. Так что в случае сомнений находите ваши ипостаси в реальном мире, и договаривайтесь о границах системы.

Теперь нужно разобраться, что такое компоненты, модули, размещения — типовые именно для инженерии ипостаси системы.

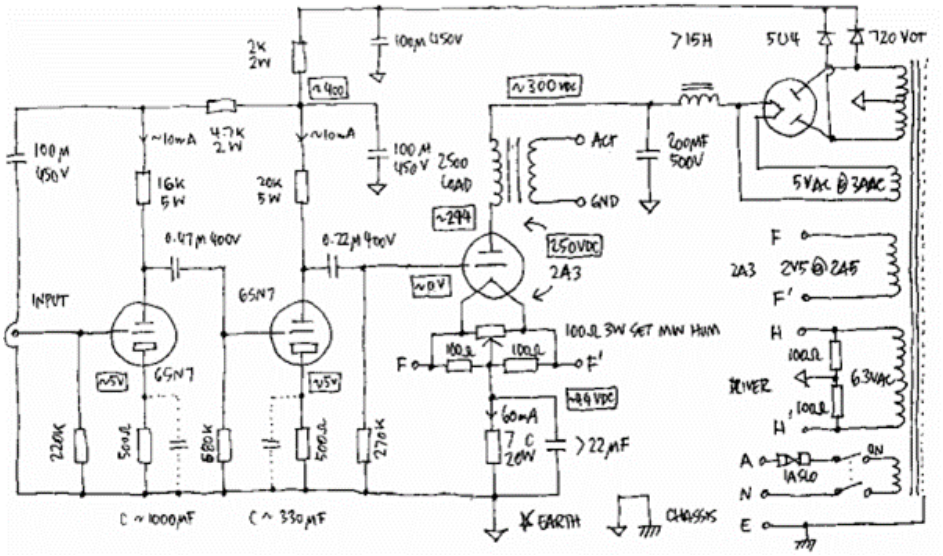
### *Компоненты*

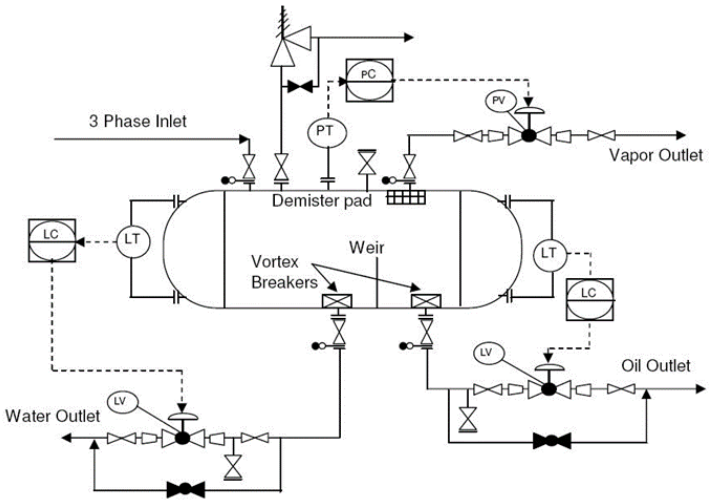
В современном русском языке компонента/компонент примерно одинаково по частоте употребляется как в женском роде, так и в мужском. Никакой нормы тут нет, так что не нужно обращать на это внимания: как привыкли писать и произносить, так и продолжайте. Вас поймут и не поругают.

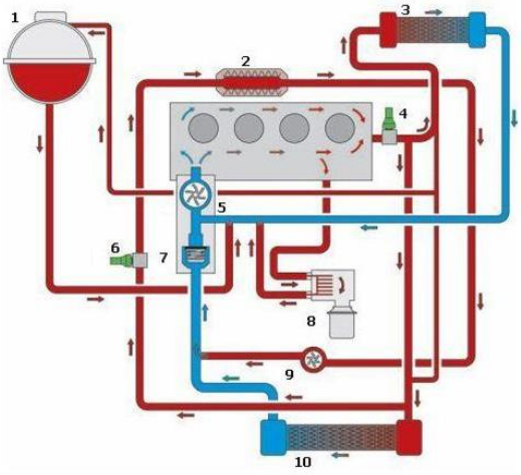
Компоненты — это ипостаси системы, в которых она выполняет какую-то функцию (имеет какое-то назначение, предназначение) в составе надсистемы/использующей системы. Помним, о том, что слово “функция” имеет множество самых разных значений! Но чаще всего функция — это поведение (т.е. описывается глаголами или отглагольными существительными — “переноска”, “охлаждение”, “освещение”).

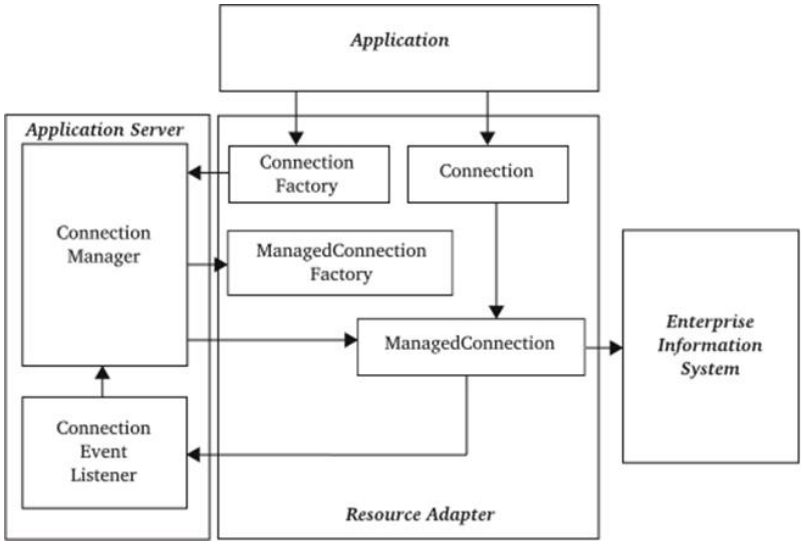
Чаще всего система определяется именно как компонента: нечто, имеющее своё назначение в объемлющей (использующей, над) системе и взаимодействующая с другими частями этой объемлющей системы для получения эмерджентности (функции надсистемы по отношению к над-над-системе). Компоненты нужны для показа взаимодействия частей системы и объяснения того, как система работает. Тем самым система в своей ипостаси компоненты представляется во время *функционирования* (run-time, operation).

Компоненты часто называют “компонеты и соединения” (components and connectors), ибо для показа связи компонент между собой используются соединения между портами (ports) компонент — и не путайте “соединения” с “интерфейсами” модулей. Диаграммы, показывающие соединения компонент друг с другом и удобные для объяснения принципа функционирования системы (”как работает система”), чаще всего называют “принципиальными схемами”:






Если для каждой компоненты ещё можно представить какой-то физический продукт (модуль), то для соединений эти модули не так очевидны: в радиосхеме соединения между компонентами не соответствуют обычно индивидуальным проводам (например, соединение может идти и через шасси), равно как в гидравлических схемах одно соединение необязательно соответствует одной трубе.

Важно, что все эти "схемы" часто не предполагают какого-то сорта физического (продуктного, модульного) воплощения в материальном мире, это остаётся для

других описаний, более удобных для этого — хотя о компонентах и говорят, как о физических элементах, но не уточняют конструкцию. Так, для каждого резистора на принципиальной схеме продуктное описание можно найти где-то в спецификации, на монтажных чертежах (печатной платы, например). Для каждого насоса в P&ID диаграммах про "физические" насосы-как-продукты больше можно узнать из закупочных спецификаций или 3D-модели. На самих же диаграммах приведены "функциональные" объекты (компоненты) и связи между ними. Компонентные диаграммы можно также найти в современных средствах мультифизического моделирования (Modelica). Все эти "клиент-серверные", "клиент-middleware-сервер" подстили описаний софтовых систем — это примеры компонентного стиля описаний, "как оно работает".

По принципиальным схемам чаще всего проводится мультифизическое моделирование: рассчитываются режимы работы, оптимальные характеристики отдельных компонентов и т.д.

Философски очень похоже, что компонента это “действующее лицо” в системе, а модули назначаются на соответствующие роли в качестве “исполнителей”. Поэтому замена какого-нибудь компонента-насоса на атомной станции, представленного модулем одного производителя на модуль другого производителя описыватся практически идентично замене президента Клинтона на Буша. Также принято говорить о компонентах не как о ролях, а как о реальных физических объектах.

В стандарте IEC 81346 принято давать обозначения компонентам (определяемым там как функциональный аспект объекта), начинающиеся с префикса = (таким образом сопротивление на радиосхеме может быть обозначено =R1).

Конечно, нужно помнить, что разные описания “гибридны” — люди на схемах вполне могут уточнить, что функция повышения давления жидкости реализуется модулем-насосом, а не перепадом высот и модулем-жёлобом, или даже явно указать марку и технические характеристики оборудования, которое нужно закупить!

### *Модули*

Модуль — это элемент конструкции, продукт, сборочная единица, физический объект. Это “исполнитель” роли компоненты. Модули обсуждаются, когда необходимо разобраться со временем разработки системы: как зависят модули системы друг от друга в плане разработки и изготовления (если в одном из модулей при разработке и/или изготовлении допущена ошибка, и это приводит к ошибочной работе другого модуля, то этот другой модуль называется зависящим от первого). Модуль так и определяется: что-то, что является результатом работы. Дырка — это модуль (ибо дырку нужно просверлить), сварной шов тоже модуль (ибо это результат работы по сварке). Функции модуля в системе не обсуждаются: модуль-микроскоп в некоторых системах колет модули-орехи, а в некоторых системах служит пылесборником.

Проще всего понять разницу между модулем и компонентом можно на примере двухмоторного самолёта: двигатель самолёта — это модуль, он разрабатывается один раз. А вот левый и правый двигатели — это компоненты, самолёту для полёта нужны оба двигателя, иначе он не выполнит своей функции.

Модули связываются друг с другом через “интерфейсы”, которые прежде всего рассматриваются с точки зрения их “видимости” (доступности для подключения модулей друг к другу). Зависимый модуль имеет слот для подключения того “низкоуровневого” модуля, от которого он зависит.

Лампочка как модуль домашней электрической системы имеет цокольный интерфейс, домовая проводка имеет цокольный слот, куда вкручивается лампочка. Если лампочка не будет работать (например, не подойдёт по интерфейсу, или будет слишком ярко светить), то зависимая от неё домашняя электрическая система тоже не будет работать. Воткнуть лампочку сразу в городскую электросеть, минуя домашнюю электросистему нельзя, её интерфейс недоступен (”не видим”) в городских низковольтных электрических системах, и уж тем более в высоковольтных электрических системах странового масштаба.

Модули взаимозаменяемы на своих интерфейсах, поэтому одну и ту же компоненту могут играть несколько разных модулей — лампочка может быть накаливания, может быть на светодиодах.

Диаграммы зависимостей (например, вводящие “слои” модульной структуры, где каждый слой зависит от другого) в описании модулей так же часты, как и просто перечисляющие модули списки. Для описания модулей часто используют и чертежи, часто модули описывают в терминах их интерфейсов, интерфейсы часто определяют стандартами.

Вот модуль “FR160B PCB 2-Layer USB Portable Power Module — Green (3.5 x 2.6 x 1.5cm)”:

Обратите внимание, в описании модуля прежде всего приводятся его интерфейсы (но не соединения, ибо это рассмотрения модуля, а не компоненты!): USB Universal Serial Bus) и размеры для размещения в корпусе. Приводятся также сведения по изготовлению (PCB 2-Layer, двухслойная печатная плата). Понять из описания модуля, как он работает или как будет работать система, в слот которой модуль включается, обычно нельзя — но зато можно понять, как подключить модуль/собрать систему, как поделить работу между конструкторами/проектантами, как изготовить модуль.

IEC 81346 перед обозначением модуля ставится дефис: -R1 (там это называется “продуктный аспект объекта”).

### *Размещения*

Мир трёхмерен, и каждая система характеризуется ещё и своим местом в пространстве. Одна из ипостасей системы — это место, которое она занимает.

Например, насос может быть компонентой (функция “обеспечить подъем давления”, номер P-101 на гидравлической схеме), может быть модулем (насос НС-1234 Пензенского завода, серийный номер №56789), но может быть и лежащим на складском месте №159 склада №689, ибо ещё не выдан в монтаж. Это разные ипостаси одной и той же системы.

В инженерном проекте строительства атомной станции минимальное число разбиений 12 (в том числе систему разбивают на части не только соответствующие каким-то функциональным, конструктивным или топологическим особенностям

системы, но и по самым разным другим принципам. Например, в системе-здании/сооружении выделяют захватки — часть здания или сооружения с одинаково повторяющимися комплексами строительных процессов выполняемыми каждый в отдельности с ритмом определенно равномерного времени (например, какую часть системы-сооружения может сделать бригада бетонщиков, или арматурщиков, или штукатуров за смену или неделю — целое здание, этаж, квартиру, 100м2 и т.д.). В случае необходимости бригады делят на звенья (1-5 человек), а часть захватки при этом, выделяемой одному звену, называют делянкой. Это всё способы поделить систему на части! Захватки и делянки — это варианты ипостаси/аспекта размещения (location, места). Об этих местах можно говорить так же, как о физических объектах (как об модулях, аналогично тому, как говорят о компонентах как о физических объектах). См., например, обсуждение захваток и делянок в работе каменщиков — http://gardenweb.ru/ponyatie-o-delyankakh-i-zakhvatkakh, при монтаже автокраном — http://korica.info/2012/10/razbivka-zdaniya-na-montazhnye-zaxvatki.html

Размещения тесно связаны с логистическим аспектом инженерного проекта, они крайне важны для менеджеров (планы работ часто привязываются именно к размещениям).

В IEC 81346 для обозначения размещений принят префикс "+".

*Структура системы: разбиения.*

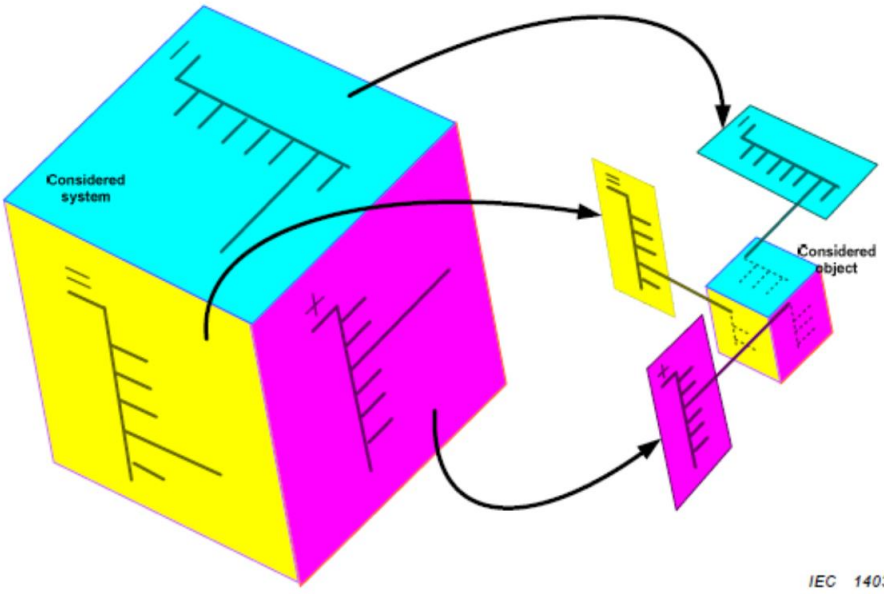
Разбиения (breakdowns)

Система сама входит частью в надсистему и состоит из частей-подсистем (или частей-элементов, если нет желания далее рассматривать структуру этих частей). Холархия (иерархия холонов) связывает объекты отношениями часть-целое (part_of relations), они же отношения "сборки" (composition) или "разборки" (decomposition), они же "разбиения" (breakdowns). В инженерии самые разные разбиения чрезвычайно распространены:

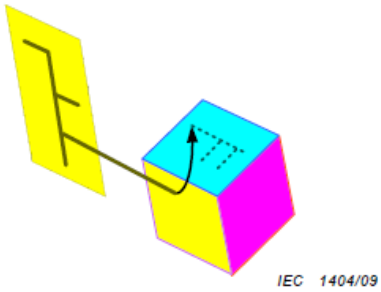
- Функциональное разбиение (functional breakdown structure), чаще называется functional decomposition
- Разбиение работ (work breakdown structure)
- Разбиение установки (plant breakdown structure)
- Разбиение документов (document breakdown structure)
- … их множество

Важно понимать, что для разных людей интересны разные разбиения — одна и та же система может быть разбита на разные типы частей по-разному, и даже на один и тот же тип частей по-разному (смотря кто и зачем это разбиение делает). Так, ложку можно разбить на хлебало и держало. Эту же ложку можно разбить и на три части: хлебало, держало и жёсткую между ними перемычку (или гибкую перемычку, если ложка складная). Эту же ложку можно разбить на металлическую пластину, вытравленный на ней узор и выдавленную на ней вмятину. Всё зависит от того, какой деятельностью занят стейкхолдер, для которого важно то или иное разделение.

Конечно, каждая ипостась системы (компонента, модуль, размещение, и т.д.) имеет своё иерархическое разбиение на подсистемы:

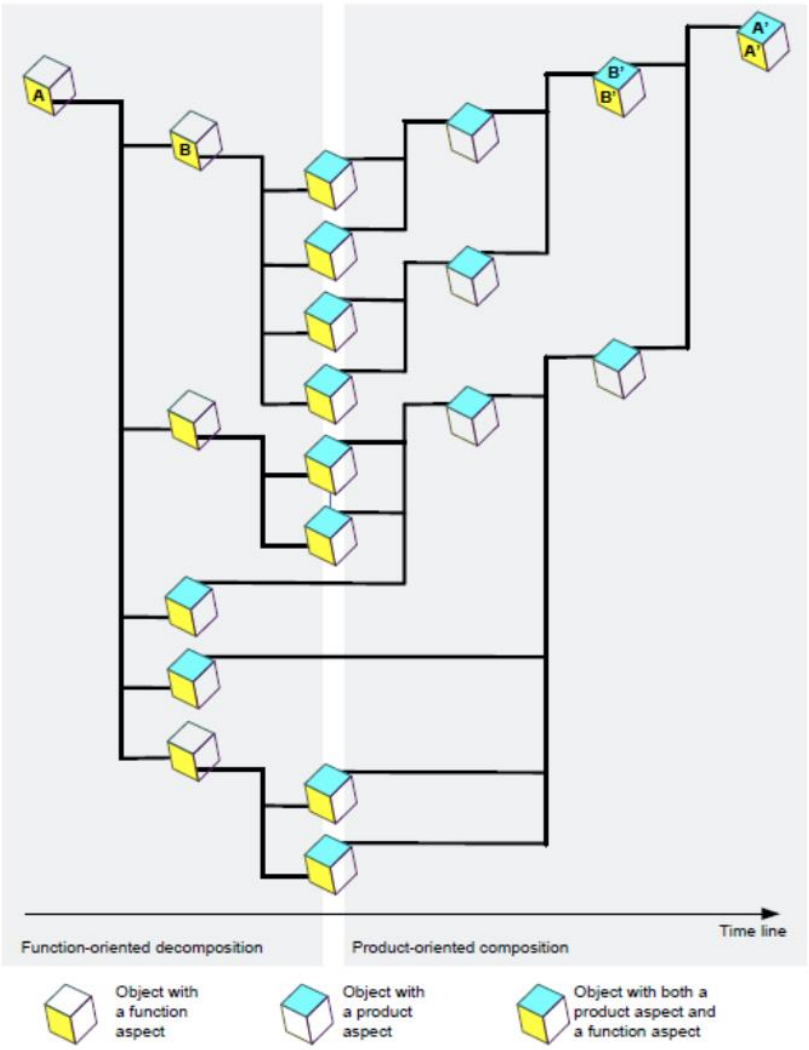



Правила структурирования из стандарта IEC 81346 (в этом стандарте компоненты представляют собой “функциональный аспект”, модули “продуктный аспект”, размещения — “аспект места”) утверждают, что структурирование технической системы базируется на аспектных разбиениях и проводятся пошагово/поуровнево — при этом выбранный аспект может меняться при каждом шаге, то есть одна ипостась системы может разбиваться на подсистемы, определяемые в другой ипостаси (например, компоненты могут разбиваться уже на модули).




Обычно шаги проводятся “сверху вниз” для компонент (декомпозиция функций компоненты) и “снизу вверх” для модулей (сборка продуктов работы в модуль). Результирующие структуры разбиений получаются не “системы вообще”, а аспектные: компонентные (функциональные) разбиения, модульные (продуктные) разбиения и разбиения размещения (места).

При проектировании функциональное разбиение “сверху вниз” обычно заканчивается продуктным разбиением “снизу вверх” — т.е. начиная с какого-то уровня компонент становится понятным, какие модули будут выполнять функции этих компонент. Обычно при таком подходе все подсистемы нижнего уровня имеют указанными оба аспекта (т.е. они представлены в обеих ипостасях — и компоненты, и модуля), то же относится к некоторым высокоуровневым подсистемам (IEC 1392/09):




A’ и B’ означают, что определение и компоненты и модуля могут изменяться, когда мы назначаем модули на компоненты, совмещаем понимание соотнесения функций на элементы конструкции (архитектурное проектирование именно в этом и заключается: совмещение компонентной и модульной структур). Например, у вас нет требуемого для компоненты-резистора на 4КОм принципиальной схемой модуля-резистора на 4КОм, но есть резистор на 2.2КОм — вам может захотеться изменить компоненты в принципиальной схеме, чтобы использовать наличные модули. Или наоборот, вы не соглашаетесь менять принципиальную схему — и тогда вам придётся как-то изготовить резистор-модуль на 4КОм (например, поставив последовательно два резистора-модуля на 2.2КОм как составной модуль и проверив, что 4.4КОм уложатся в требуемые принципиальной схемой пределы для резистора-компоненты).

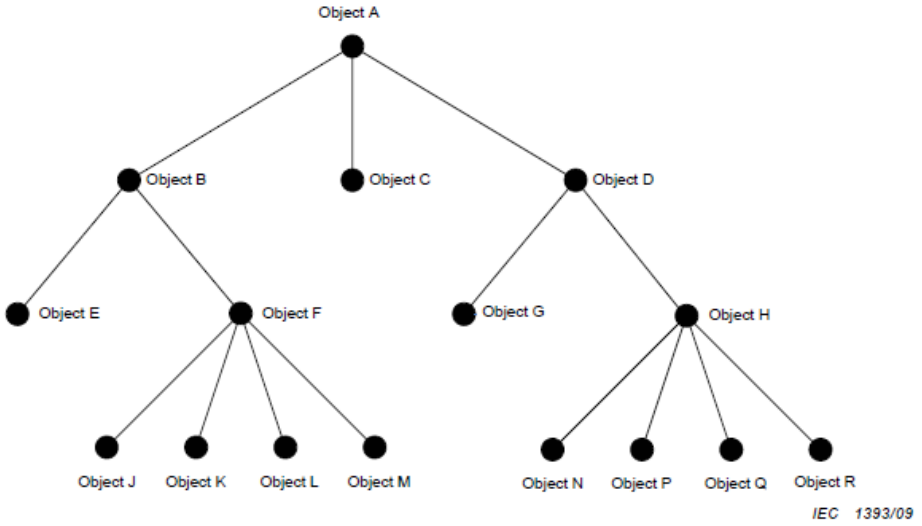
В большинстве "железных" систем модули и компоненты соответствуют друг другу 1:1, что позволяет произвольно их путать. Но это может приводить к путанице в тех редких случаях, когда этого соответствия нет. Так, описания "платформ" (слои) в терминах компонент не хороши. "Платформы" нам важны для того, чтобы проще спроектировать и собрать систему (т.е. это описание времени разработки-изготовления, а не времени эксплуатации). А компоненты важны, ибо по большому счёту работоспособность системы зависит именно от наличия связанных и работающих компонент целевой системы, выполняющих свои функции и дающие эмерджентную функцию целевой системы.

Обратите внимание, что диаграмма совмещения компонент/функций и

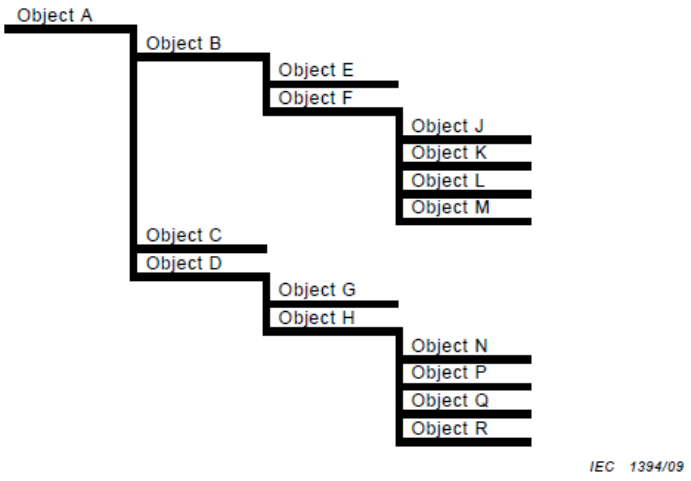
модулей/конструкции как-то напоминает латинскую букву V – это не случайно, V-диаграмма (разделяющая жизненный цикл на практики определения системы и практики воплощения системы – левую и правую ветвь V) в системной инженерии является одной из центральных. Мы ещё неоднократно встретим варианты V-диаграммы.

### Представления разбиений

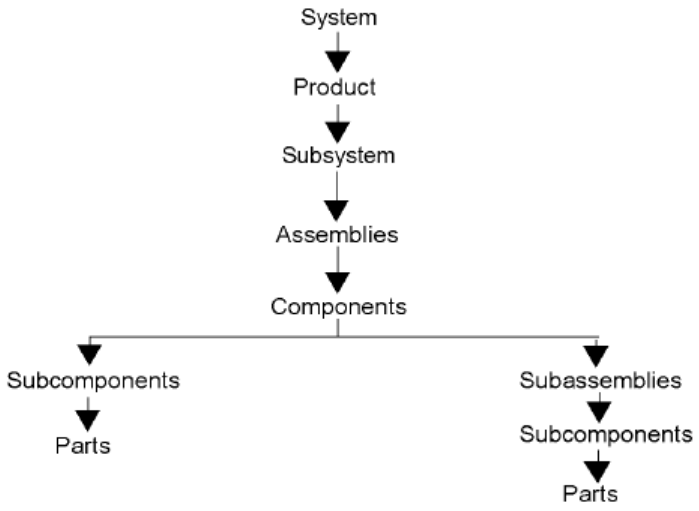
Разбиения представляются деревьями в разной графической форме (структурными диаграммами). Чаще применяется не традиционное "математическое дерево":




Но "дерево лесенкой" (которое удобней отображать в различных инженерных информационных системах):




Старинные инженерные стандарты предписывали конкретное число уровней и названия уровней разбиений. Так, ISO 26702 определяет следующие уровни с чётко определёнными названиями, при этом аспектность по факту игнорируется:




Elements of the system may include hardware, software, and humans dependent on the system definition.

**Figure 1—Hierarchy of elements within a system**

Ещё один старинный набор стандартов ЕСКД (ГОСТы Единой Системы Конструкторской Документации) делает то же самое жесткое определение, но эти уровни другие:

- Деталь — это изделие, выполненное из однородного материала без применения сборочных операций (например, болт, гайка, вал, втулка).
- Сборочная единица — изделие, составные части которого соединяются между собой в процессе сборки с помощью резьбы, пайки, сварки и т. п. (например, редуктор).
- Комплекс — два и более изделия, предназначенных для выполнения взаимосвязанных функций (например, технологическая линия).
- Комплект — это набор из двух и более изделий, имеющих общее эксплуатационное назначение вспомогательного характера (комплект запасных частей, комплект инструментов и принадлежностей).

Обратите внимание, в ЕСКД никакого системного подхода нет ни “по названиям”, ни “по сути”. Кроме “конструкторской документации” единые системы стандартов ГОСТ есть и для строительной документации, программной документации и т.д. Поэтому в сложных системах структурирование частей системы по уровням “часть-целое” и принципы обозначения элементов системы крайне разнятся.

### *Обозначения систем*

Задание обозначений для каждой части системы тесно связано с принципами (структурного, обычно по отношениям “часть-целое”) разбиения системы. Задание обозначений определяется стандартами — международными, национальными, отраслевыми, предприятия. Часто обозначения называют “кодами”.

Так, в российском машиностроении принята система («система» тут из систематики!) обозначений ЕСКД (используются “коды ЕСКД”), а в строительстве российских атомных станций сейчас чаще всего используется система обозначений KKS. Проблема возникает тогда, когда какой-нибудь винтик в приборе обозначен по ЕСКД, а сам прибор обозначается по KKS (это была бы ещё самая простая ситуация, часто разных систем обозначений в сложном проекте используется добрый десяток, а в случае международного проекта и больше).

Старые системы обозначений страдают от следующих недостатков:

- Они не учитывают аспектности (т.е. выпячивают либо компоненты, либо модули, либо места, либо задают жёстко границу перехода по уровням — до какого-то уровня только компоненты, а затем только модули). Опять же, если речь идёт о какой-то узкой предметной области, то это нормально. Но как только появляется более-менее крупный проект, объединяющих продукты разных отраслей, работать (искать какие-то элементы в составе системы по их обозначениям, давать обозначения) становится трудно.
- Они позиционны, т.е. не предусматривают различного числа уровней разбиения. Это может проходить в рамках узкой предметной области — группы одинаково структурированных продуктов, но когда в проекте собираются продукты нескольких отраслей, работать становится очень трудно.

IEC 81346

Современным стандартом, описывающим структуру системы и отвечающим за обозначения структурных элементов системы в их отношениях "часть-целое" по каждому аспекту, т.е. отвечающим принципам системного подхода и поэтому подходящим для системной инженерии является IEC 81346-1:2009 "Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations. Part 1: Basic rules" — этот стандарт вобрал в себя многочисленные достижения предыдущих поколений стандартов.

Стандарт предписывает правила, по которым происходит аспектное структурирование системы и создание следующих принципам системного подхода "справочных обозначений" (reference designations — в отличие от просто "обозначений" стандарт называет "справочными обозначениями" те, которые следуют принципам системного подхода). Эти обозначения:

- учитывают аспектность
- произвольность числа уровней системы

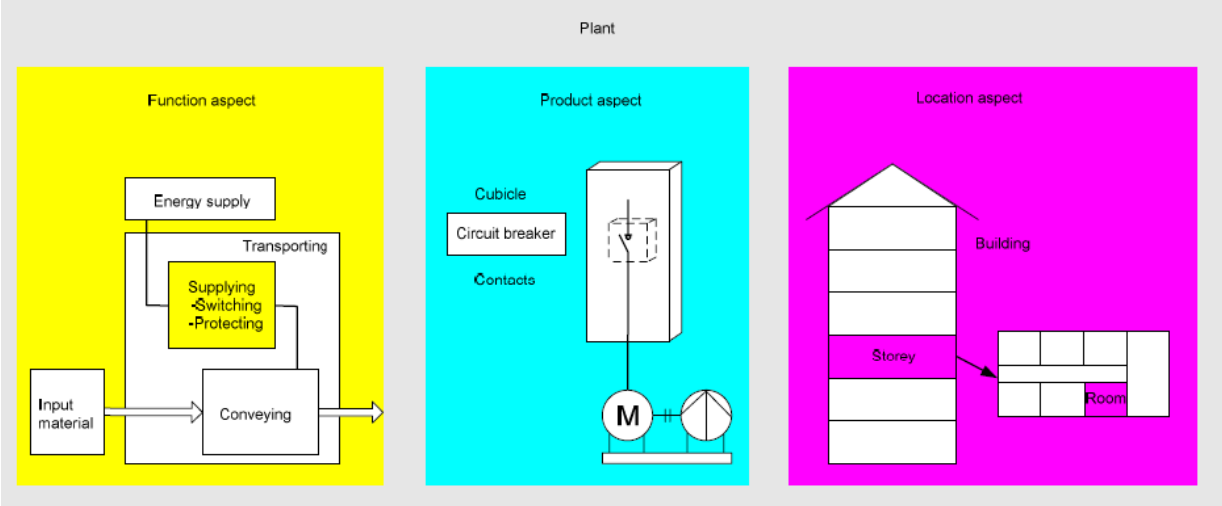
Отдельных аспектных обозначений в огромной системе могут быть миллиарды (например, в микросхеме есть 5 миллиардов транзисторов — и каждый из них должен быть проименован как компонента, как модуль, плюс ещё и размещение на кремниевой пластине. А ещё ведь нужно обозначить какие-то подсистемы, составленные из этих транзисторов! И подсистемы, составленные из этих подсистем, и так до самого "верха" — например, кремниевого чипа, или даже готовой микросхемы в корпусе. И даже дальше — контроллера, который сделан на базе этой микросхемы, системы управления с использованием этого контроллера, компрессора, который управляется этой системой, холодильной машины, куда входит компрессор, и так далее — например, до атомной станции или корабля, где используется эта холодильная машина. Не нужно и говорить, что число уровней подсистем, в которых появляется эта холодильная машина для атомной станции и корабля будут разными — это и есть возможность учёта произвольного числа уровней системы).

Вводя обозначения ("краткие имена"), нужно помнить, что для некоторых обозначений есть длинные содержательные имена, а для некоторых обозначений (каких-нибудь винтиков крепежа корпуса) таких имён даже не будет.

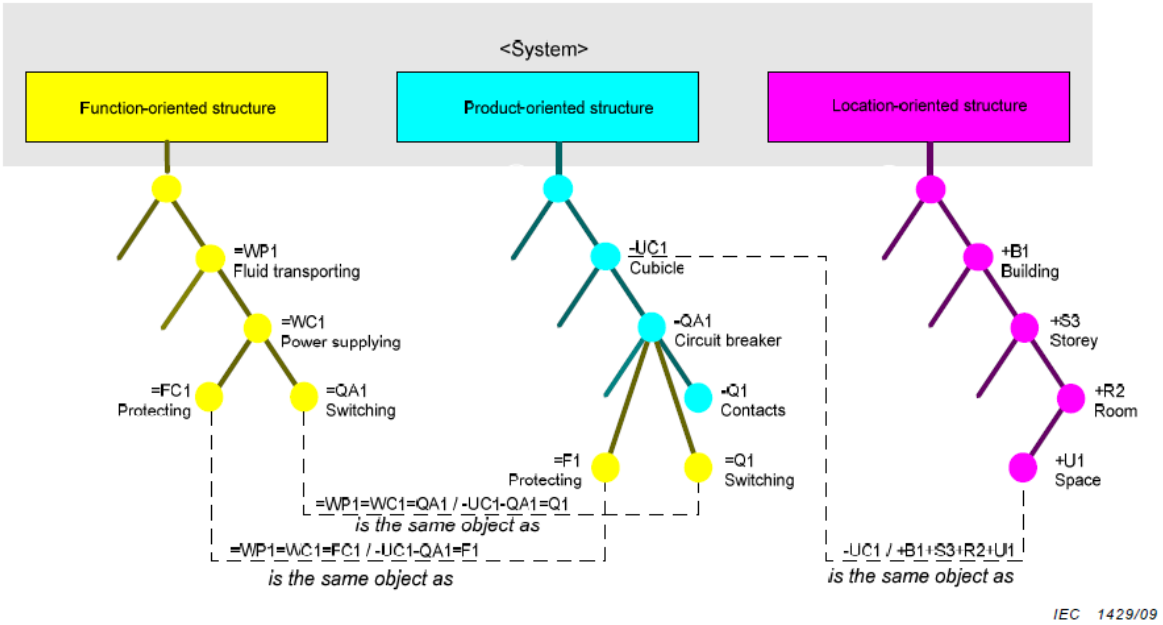
Простейший способ обозначать ипостаси систем — это давать им уникальные

порядковые номера на каждом уровне разбиений, указывая при этом спецсимволом-префиксом аспект (компонента это, модуль или размещение). Более сложный способ — это обозначения, включающие в себя какой-то (чаще всего функциональный) классификатор, т.е. отдельные символы и их сочетания, означающие принадлежность данной ипостаси к определённому классу (математическому множеству) ипостасей системы.

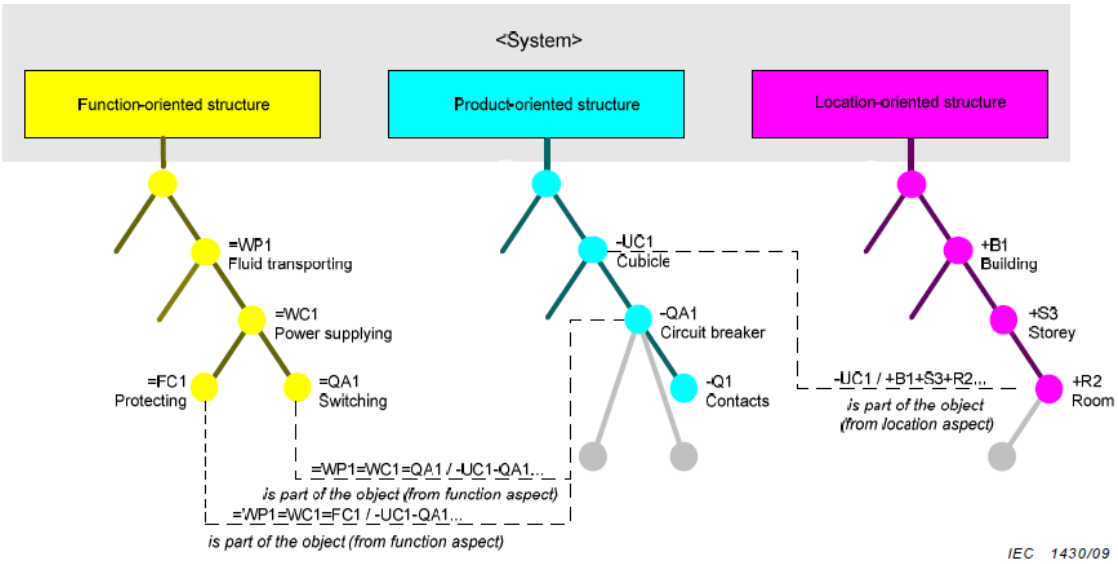
В общем случае обозначение системы состоит из цепочки обозначений какого-то маршрута по разбиению, или разных маршрутов по разным разбиениям. Например, установка (plant) по транспортировке жидкости аспектно представлена из следующих компонент, модулей и размещений (мест):




Если теперь представить разбиения и ввести аспектные обозначения каждой системы-подсистемы-элемента, то получим:




Заметим, что некоторые элементы получили полные обозначения сразу в двух аспектах. Для однозначного указания на объект достаточно иметь только одно полное имя, а остальные имена пусть указывают на “надсистему”.




Каждое обозначение — это имя холона (объекта, который состоит из частей, но сам является частью другого объекта), ибо система по определению — это холон в системной холархии (у любой системы есть её элементы-подсистемы и она является элементом в надсистеме). Поэтому все имена компонент, модулей, мест подразумевают, что перед ним будет имя надсистемы, а после него — подсистемы. Например, ...=контроллер=процессор=кэш первого уровня...

*Практики изготовления (производства)*

Почему молодых инженеров-конструкторов в конструкторских бюро мечтают принудительно на три месяца работать на завод? Потому что «проектируют такое, что потом невозможно [или очень дорого] изготовить». Вопрос: а куда нужно оправить тех, кто на заводе – ибо они не могут дёшево сделать то, что конструктор считает нужным?! В 2015 году по сравнению с 2014 годом драматически изменились практики работы с веществом. Нужно с самого начала разработки ориентироваться на современные методы воплощения системы (например, аддитивные методы, типа 3D печати), а не традиционные допотопные, знакомые по учебникам и построенным тридцать лет назад заводам.

[Замечание в сторону: я считаю, что в России нормально производить из вещества ничего нельзя, ибо будет либо очень некачественно, либо запредельно дорого по сравнению с продукцией других стран, либо и некачественно и дорого вместе. Так что мы обсуждаем ситуацию в мире, а не специфическую ситуацию в России – которая стремительно становится всё более и более специфической].

Текущая ситуация: The INCOSE Systems Engineering Handbook version 3.2.2 published in October 2011 has five percent of its content covering integration, verification, transition and validation. (цитата из http://www.incose.org/newsevents/events/details.aspx?id=203). Это косвенная мера того, сколько знаний у системных инженеров по вопросам практик изготовления. И уж тем более системные инженеры мало касаются «самого низа» V-диаграммы – получения деталей, которые будут изготавливаться.

Для ознакомления с практиками изготовления можно рекомендовать почитать:

- Про движение DevOps – результат осознавания, что воплощение важно разработчикам софта (http://theagileadmin.com/what-is-devops/, http://venturebeat.com/2013/09/30/an-idiots-guide-to-devops/).

- про advancing manufacturing, cyber-physical production systems, etc. — суперплотный обзор Carl Bass (CEO of 3D design, Autodesk) по современным изменениям в воплощении (видео): http://ailev.livejournal.com/1137544.html
- Пару прошлогодних (тут события развиваются стремительно, и материалы за пару лет уже устарели!) презентаций по цифровому производству и ссылок «вдогонку» — http://ailev.livejournal.com/1103473.html (видео и аудио по-русски)
- свежайшую книжку по 3D печати и 3D принтерах (на русском): http://3dplast.net/news/pervaya-kniga-o-3d
- материалы свежих обзоров по ссылкам тут: http://ailev.livejournal.com/1134471.html.
- тезисы по тотальной автоматизации — http://ailev.livejournal.com/1131557.html (и факультативно http://ailev.livejournal.com/1131958.html, где меньше про выход в физику – то есть меньше про *роботику*, но отвечается на ряд аргументов).

Что из этого вы готовы применить в ваших проектах? Сколько это будет стоить, если делать изготовление в России? Сколько бы это стоило, если бы вы это делали в Китае? в США? (сколько займёт времени заказ-изготовление-пересылка: перевести это замедление проекта в деньги с учётом рисков).

## 7. Определение системы: требования, архитектура, неархитектурная часть проекта

*Определения и описания*

Перед тем как систему воплотить, её нужно как-то определить (define). Это определение системы — основная альфа инженерного проекта, а выражается она рабочими продуктами, которые совместно называются описанием (description) системы. Конечно, есть огромное число вариаций этой терминологии (например, определение системы называют “проектом”, а описание — “проектной документацией”), тем не менее нужно понимать разницу: определение системы всегда есть, если есть система (если система никак не определена, то откуда вообще мы знаем, о какой системе мы говорим?!) — но вот описаний может не быть, если никто не потрудился задокументировать определение системы в рабочих продуктах.

Определение системы как альфа включает в себя основные подальфы (это только основные подальфы, их может быть много больше):

- Требования (определение системы как “чёрного ящика”)
- Архитектуру (важнейшие инженерные решения, определяющие систему как “прозрачный ящик”)
- Неархитектурная часть проекта (все инженерные решения, которые будут сочтены не самыми важными) — “рабочка”.

Продвижение в уточнении каждой из этих подальф продвигает определение системы в целом — чем больше определены требования, архитектура, неархитектурные определения, тем больше определена в целом целевая система.

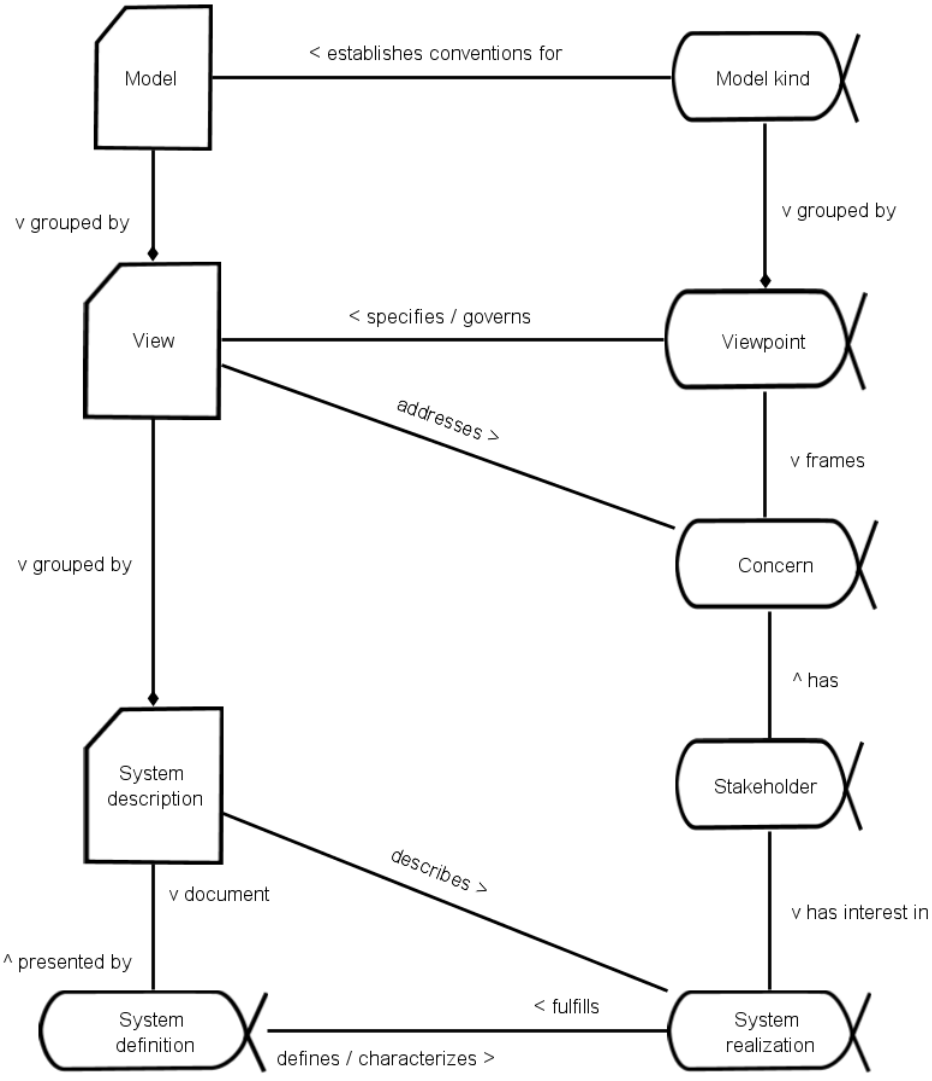
Часто выделяют проект/design как совокупность архитектурной и неархитектурных частей (вся архитектура — это design, но не весь design это архитектура). Часто

требования не считают относящимися к проекту. Иногда архитектурный проект называют “эскизным проектом”, иногда “техническим проектом”, иногда просто “проектом”, тогда описания неархитектурной части проекта называют “рабочей документацией”. А ещё есть “исполнительская документация” — это описание системы “как построено/изготовлено” (as built) в отличие от “проектной документации” (as designed).

Конечно, определений (и, соответственно, описаний) системы много больше. Так, часто в plant model (описании, или “информационной модели” установки) выделяют проектную-design модель и проектную-project модель (в русском языке это неразличимо, увы). Design часть описания — это описания функции и конструкции самой целевой системы (инженерного решения — проект в смысле инженерный проект), а project-часть — это описания предпринятия (обеспечивающей системы: работы и соответствующие планы, команда и т.д. — проект в смысле проектного управления).

### Обобщение ISO 42010 на определение системы

ISO 42010 “Systems and software engineering — Architecture description” вполне может быть обобщён с архитектурного на все виды описаний. Основные его положения можно увидеть на этой диаграмме:




Описания — это рабочие продукты, которые выражают определение системы. Обратите внимание, что определение системы на диаграмме выражено значком

альфы, а описание — значком рабочего продукта. Воплощение системы удовлетворяет определению системы, а определение системы характеризует (а пока системы нет, то определяет) воплощение системы. Тематический (т.е. финансовый, физический, архитектурный и т.д.) метод (набор практик, в своей совокупности позволяющий проводить описание по какой-то тематике) описания (vewpoint) задаёт тематическую группу описаний (veiw), а каждая группа описаний имеет свой метод описаний.

Группа описаний группирует отдельные модели (отдельные описания). Так, финансовая группа описаний может группировать баланс и отчёт о прибылях и убытках, архитектурная группа описаний может группировать компонентные, модульные и описания размещения, а также какие-то гибридные описания.

Любые группы описаний существуют только потому, что есть стейкхолдеры, заинтересованные в системе в той части, которая этими описаниями описываются. Если нет стейкхолдера, которому это описание нужно, то описания просто не должно быть — зачем трудиться для создания того, что не будет никем использовано? Если заинтересованный стейкхолдер есть, но нет нужного ему описания — большая вероятность, что интересы стейкхолдера не будут учтены (помним, что успешной системой называется та, которая удовлетворяет стейкхолдеров. Не будет нужного стейкхолдеру описания — будет огромный риск создания неуспешной системы).

Методы описаний поставляются наукой. Дисциплины, задающие виды моделей — это научные (учебные, профессиональные) дисциплины, предлагающие различные виды моделей. Модель (тут мы чаще всего будем говорить об информационной модели, а не о физической "натурной" модели) — это система M, которая может быть использована в каких-то пределах для получения знания о системе S. Модель в разы, разы и разы проще реальной системы S, она опускает неполезную для интереса стейкхолдера информацию, и предоставляет полезную.

Важно запомнить, что нельзя что-то описать, если не знаешь метода описания. Метод описания включает какую-то теорию (дисциплину — позволяет определять систему в терминах каких-то альф), инструменты (чаще всего программные средства), форматы рабочих продуктов описаний (модели данных для баз данных, реквизиты документов, системы обозначений для диаграмм), какие-то советы для успешного составления описаний. Метод описания может быть

- библиотечный (library) в переносном смысле слова — описанный в какой-то литературе, и на это описание можно сослаться. Иногда библиотечные методы описания называют Framework (и даже library framework) — например, TOGAF (The Open Group Architecture Framework) это "метод описания архитектуры консорциума по стандартизации The Open Group", i* requirements framework — метод описания требований i*.
- Собственной разработки (что по факту означает необходимость проведения научной работы: изобретения каких-то абстрактных сущностей, постоения какой-то новой теории, разработки новой нотации, а затем документирования результатов этой работы).

Нужно запомнить: описаний без метода описания не существует. Если вы пишете прозой, то метод описания — "проза, жанр эссе". Если вы описываете финансы (а хоть и финансовый баланс), то вы должны указать метод: российская система бухучёта или международная. Вы всегда явно должны давать ссылку на метод описания, по возможности его уточняя. А если используется метод описания

собственной разработки, а не библиотечные описания, то необходимо в состав проектной документации (описания системы) включать описание метода описания.

Очень часто метод описания известен неявно (например, машиностроительное предприятие будет использовать "чертежи по ЕСКД"). Но стоит этим чертежам попасть в иностранную фирму (или строительную организацию, привыкшую к "чертежам по СПДС"), и содержание этих чертежей становится уже не таким понятным. Поэтому явное указание метода описания — это хороший тон в инженерных проектах. Если вы приводите диаграммы сценариев использования (use case diagrams) для описания требований, то обязательно скажите, что это use case diagrams (а не придуманный вами самими способ задавать функциональность системы), а ещё лучше — дайте ссылку на ту книгу, из которой вы взяли конкретный использованный вами метод описания и сам вид этих диаграмм.

Нужно особо отметить, что описания — это всегда описания какой-то системы в системной иерархии (холархии). Имя описания может добавляться к имени системы через & (например, =процессор&логическая схема или -насос&инструкция по подключению).

### Контроль конфигурации

Конфигурация системы — это то, из чего на какой-то момент времени состоит система. Конфигурация находится под контролем, если конфигурация определения системы соответствует конфигурации воплощения системы. Если какие-то части этих конфигураций не соответствуют друг другу, то говорят о конфигурационных коллизиях (например, если на принципиальной схеме изображено 12 насосов, а в реальной системе их стоит 11). В какой-то момент в США была отобрана лицензия атомной электростанции: из-за многочисленных модификаций оборудования чертежи перестали соответствовать реально установленному оборудованию, и эксплуатация станции в такой ситуации была признана небезопасной.

Контроль конфигурации обеспечивается тем, что конфигурация может быть собрана в любой момент. В частности, это означает необходимость знания о том, где лежат отдельные описания, а также необходимость контролирования разных версий отдельных частей описания: если в проект попали "чертежи с исправлениями после совещания 2 мартобря" и одновременно "чертежи, куда не успели внести исправления после совещания 2 мартобря", то ничего хорошего от использования такого проекта ждать не приходится.

Для того, чтобы обеспечить контроль за конфигурацией (configuration control), используется практика управления конфигурацией (configuration management). Эта практика системноинженерного менеджмента — она занимается поддержанием целостности системы на протяжении всего жизненного цикла.

Заметим, что сама практика управления конфигурацией включает получение различных описаний. Так, именно в рамках этой практики выпускаются различные виды спецификаций закупаемого/изготавливаемого оборудования/изделий — BOM (bill of materials, список комплектующих).

Для управления конфигурацией определения системы сейчас используют информационные системы управления версиями (в программной инженерии), PDM (product data management — системы хранения информации проекта-design), PLM (product life cycle management: это PDM+система управления изменениями и поддержка интерфейсов в другие информационные системы других стадий жизненного цикла — системы закупок, например), EAM (enterprise asset

management, система управления активами — используется для учёта установленного оборудования стадии эксплуатации).

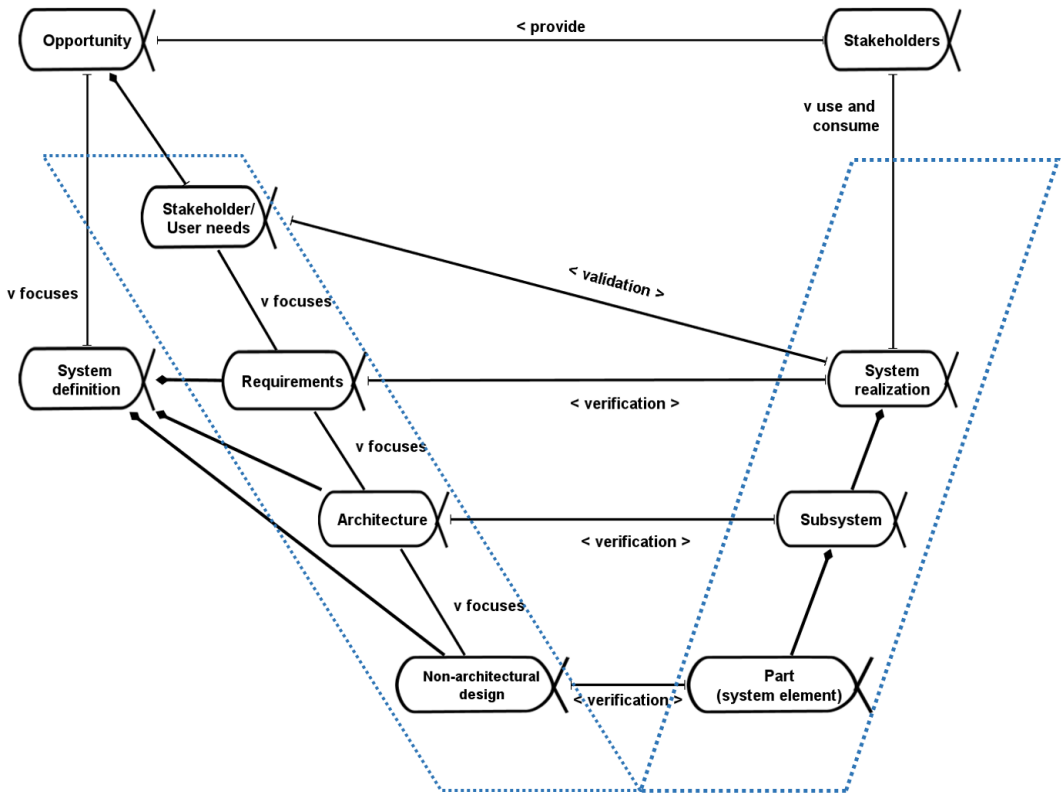
Контроль конфигурации довольно сложен, ибо в больших системах непрерывно меняется как определение, так и воплощение системы.

Подробней про управление конфигурацией в управлении жизненным циклом см. ряд материалов:

- http://ailev.livejournal.com/1005515.html — системная инженерия, инженерный менеджмент и инженерные сервисы
- http://ailev.livejournal.com/933073.html — управление конфигурацией
- http://ailev.livejournal.com/929655.html — управление жизненным циклом
- http://ailev.livejournal.com/939303.html — функции систем управления жизненным циклом.
- http://ailev.livejournal.com/1058803.html — федерирование инженерных информационных систем разных стадий жизненного цикла (этот текст в существенной мере относится к практике управления информацией, необходимой для надлежащего контроля конфигурации. Он требует для своего понимания профессионального знания моделирования данных).

Важнейшую роль в управлении конфигурацией играет практика обозначений систем (чаще говорят «практика кодирования»), следующая разбиравшемуся несколько страниц назад IEC 81346.

Фокусирование определений системы




Это ещё один вариант изображения V-диаграммы, использующейся в системной инженерии для демонстрации разбиения практик жизненного цикла системы на практики её определения и воплощения (левая и правая ветви V), при этом

перекладины символизируют соответствие определения и воплощения друг другу (должно быть так: «что определяли, то и воплотили» -- это обеспечивают практики проверки и приёмки).

Говорят, что потребности стейкхолдеров фокусируют требования, требования фокусируют архитектуру, архитектура фокусирует неархитектурную часть проекта («рабочую документацию», «рабочку»). Это нужно понимать так, что из свободы творить что угодно (свободы инженерных решений) потребности стейкхолдеров оставляют только то, что нужно стейкхолдерам. Если стейкхолдерам нужно лазерное медицинское оборудование, то вряд ли им нужны требования для автомобиля. Если требования определяют автомобиль, то архитектура должна быть именно автомобиля — и не расплываться, архитектура должна быть сфокусирована на автомобиле, свобода архитектора в выборе инженерных решений на каждом уровне фокусирования уменьшается. Минимальная свобода — у конструктора или проектанта, который разрабатывает рабочую документацию (детальные чертежи, по которым будет изготавливаться изделие или сооружаться сложный инженерный объект типа атомной электростанции). Каждое определение системы ограничивает/фокусирует последующее: определяемая требованиями функция отвечает нуждам стейкхолдеров — не больше и не меньше, определяемая архитектурой структура системы отвечает функции — не больше и не меньше, определяемая неархитектурными (рабочими) инженерными решениями часть проекта отвечает архитектуре, технологическая документация (определяющая технологию изготовления — какие инструменты, обработки этими инструментами и последовательность операций по изготовлению и сборке нужно использовать для воплощения системы) входит в рабочую документацию и подчинена рабочим инженерным решениям.

Но “отвечать” и “фокусироваться” вовсе не означает последовательности во времени: так, ограничения в технологии (например, политика чучхе — “опоры на собственные силы”, ограничивающая импорт нужных инструментов) вполне могут повлиять на архитектурные решения, или даже на саму возможность выполнить требования — и тогда получаемая в конце цепочки фокусирования новая информация заставляет менять определения, вплоть до изменения требований или признания невозможности инженерного проекта.

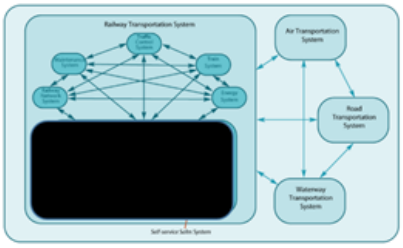
### Практики проверки и приёмки

Проверка (verification) и приёмка (validation) — это важнейшие практики системной инженерии. Важно отличать их друг от друга.

Проверка — это проверка того, что система соответствует какому-то описанию целевой системы (требованиям, архитектуре, неархитектурной части проекта). Приёмка — это проверка того, что использующая система соответствует желаемому стейкхолдером её описанию, если в её составе работает целевая система. Это не так хитро, если поглядить на диаграммки:

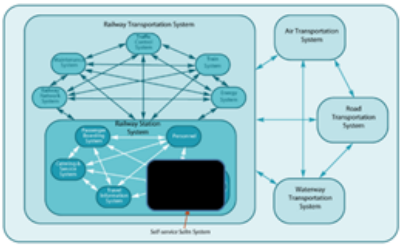
**Stakeholders/user needs define Using system as a black box that includes System of interest and Systems in operation environment**

**Validation: using system fit stakeholders/user needs**



**Requirements define system of interest as a black box that includes subsystems**

**Verification: system of interest fit requirements**



**Architecture define system of interest as a white (transparent) box with a subsystems**

**Verification: subsystems fit architecture**

Очень частая ошибка: забывают о необходимости валидации. В результате система удовлетворяет инженеров, которые эту систему сделали в соответствии со спецификациями, но не удовлетворяет стейкхолдеров: испытания должны проводиться не только для целевой системы (проверки), но и для использующей системы (приёмка!). Только после приёмки (то есть удостоверения в том, что использующая система работает как надо) системные инженеры могут считать, что их работа выполнена.

Системные инженеры должны обязательно обеспечивать оправдание ожиданий стейкхолдеров не в том, что целевая система работает! Нет, они должны оправдывать ожидания стейкхолдеров в том, что работает использующая система, в составе которой задействована (наряду с другими системами в её операционном окружении) целевая система.

Вузовский курс проверки и приёмки вполне объёмист. Вот, например, 3 кредитный курс US SanDiego этого года http://extension.ucsd.edu/studyarea/index.cfm?vAction=singleCourse&vCourse=BUSA-40414, или всё то же самое в компактном режиме производственного двухдневного тренинга — http://www.tonex.com/training-courses/system-verification-validation/. Product integrated verification and validation – 200 страниц слайдов для «военных системных инженеров»: http://www.academia.edu/1745073/Systems_Engineering_MS_Course_Integrated_Product_Verification_and_Validation

Главное – это понять наличие двух разных дисциплин: проверка (мы проверяем, все ли шестерёнки в наших часах крутятся так, чтобы показывать время) и приёмка (проверка того, что пользователь может узнавать время по нашим часам). То есть проверка целевой системы – правильно ли мы её создали и собрали. И проверка использующей системы – правильно ли она работает, имея нашу целевую систему как составную часть.

Обычно для больших систем (авиация, космос, атомная промышленность, медицина) проверка и приёмка жёстко регламентируются «обязательными стандартами». Если велика угроза возврата продукции (серийный выпуск изделий), то есть огромный стимул проверять продукт перед продажей – там тоже уровень проверки довольно высок. В остальных случаях проверка и приёмка проходят «по вкусу» (а о вкусах, понятно, не спорят), в мерах приверженности участников разработки тем или иным практикам.

Основной тренд – это непрерывное автоматизированное разноуровневое

тестирование, на всех этапах жизненного цикла, в том числе в ходе эксплуатации. Сегодня даже принтер может напечатать тестовую страницу в любой момент времени, не только в момент заводских испытаний. Раньше автомобиль мог быть протестирован только на станции обслуживания, сейчас автомобиль по максимуму тестирует себя сам (ибо весь напичкан датчиками и контроллерами).

Практики проверки и приёмки приходят сейчас главным образом из software engineering (как обычно), а потом медленно с лагом 10 лет принимаются системной инженерией. Поглядите на http://en.wikipedia.org/wiki/Portal:Software_testing (там внизу страницы topics, вам неплохо бы понимать, что скрывается за этими словами). Это хорошая точка для разбирательства с предметной областью.

Очень интересно смотреть на то, что происходит в плане автоматизации испытаний: ибо для того, чтобы что-то автоматизировать, нужно глубоко изучить предметную область. Размахивание руками не подходит как способ объяснить «автоматизаторам», что же им нужно сделать – приходится для этого как-то структурировать предметную область V&V.

**Что нужно запомнить: на V&V (проверки/приёмки/контроль/тестирование/испытания) приходится до половины всех затрат на разработку!**

Когда речь идёт об «аналитике» (человеке, который ни на что не влияет, но систему описывает), то он не заботится об испытаниях сделанного по его описаниям. Инженер, если описывает систему, то заботится о том, чтобы проверить сделанное по этим описаниям, он не «аналитик», он «синтетик». Когда «аналитик» что-то описывает, игнорируя испытания/тестирование, то он вполне может ошибиться вдвое в части определения объёма работ и стоимости проекта.

Вот некоторое количество ссылок на материалы по проверкам и приёмкам:

- Классика жанра (классические определения и общее понимание предмета):
  - место проверки и приёмки в ходе воплощения системы (водопад): http://sebokwiki.org/wiki/System_Realization
  - проверка: http://sebokwiki.org/wiki/System_Verification
  - приёмка: http://sebokwiki.org/wiki/System_Validation
  - что делает инженер по испытаниям (hardware test engineer): http://en.wikipedia.org/wiki/Test_engineer
  - чему учат: пример программы курса — http://www.nafems.org/events/nafems/2014/vandv3/, ещё вариант программы — http://www.tonex.com/training-courses/system-verification-validation/, а вот пример материала (слайдов) для вузовского курса — http://www.academia.edu/1745073/Systems_Engineering_MS_Course_Integrated_Product_Verification_and_Validation).
  - слайды занятия: что обсуждается в «тестовых стратегиях» (и чуть-чуть про тестирование роботов): http://courses.csail.mit.edu/6.141/spring2013/pub/lectures/Lec07-SystemEngineering.pdf
- TDD, test driven development (разработка софта, вариант agile методологии разработки):
  - driven против based — http://ailev.livejournal.com/1015692.html

    - тестоориентированная разработка: http://en.wikipedia.org/wiki/Test-driven_development
    - пример, как и в пользу чего отказываются от TDD (читать нужно так «TDD очень хорош, но это явно не панацея в тестировании»: http://david.heinemeierhansson.com/2014/tdd-is-dead-long-live-testing.html
    - что ещё кроме TDD бывает в agile (а именно, developer TDD, developer regression testing, reviews/inspections, end of lifecycle testing, acceptance TDD, parallel independent testing, independent regression testing): http://www.ambysoft.com/essays/agileTesting.html
- Model-based testing (и по сопричастности test automation): тут очень интересно, ибо приходится разбираться, как устроена предметная область – т.е. что такое испытания как таковые.
    - model-based testing: http://en.wikipedia.org/wiki/Model-based_testing
    - пример слайдов занятия по model-based testing: https://www.st.cs.uni-saarland.de/edu/testingdebugging10/slides/19_ModelBasedTesting.pdf
    - time partition testing: http://en.wikipedia.org/wiki/TPT_%28software%29
    - обезьянки против роботов (на русском): http://penxy.com/joso, http://penxy.com/wylo, http://penxy.com/rygi.
- Test of Cyber-physical systems (control system engineering):
    - разница в Model in loop(MIL), Software in loop(SIL), Processor in loop(PIL), Hardware in loop(HIL): https://www.linkedin.com/groups/Difference-between-MIL-SIL-PIL-109866.S.167229094
    - hardware-in-the-loop: http://home.hit.no/~hansha/documents/lab/Lab%20Work/HIL%20Simulation/Background/Introduction%20to%20HIL%20Simulation.pdf
    - примеры испытательных стендов (как «что-то-in-the-loop»), можно начать с http://www.autonomie.net/projects/engine_loop_18.html (и там есть ещё коротенькие определения терминов в картинках — http://www.autonomie.net/references/model_based_design_defs_24.html ).
- Моделирование инженерных обоснований (в том числе assurance case, его посмотрите обязательно) —http://ailev.livejournal.com/811715.html (и там много-много ссылок внутри).

Как всегда, основной тренд в системной инженерии – это сдвиг всех проверок и приёмок влево на V-диаграмме (см. свежий блог пост Donald Firesmith – «Four Types of Shift Left Testing», http://blog.sei.cmu.edu/post.cfm/four-types-shift-left-testing-082).

Упражнение: какая стратегия тестирования/испытаний (проверок и приёмки) в вашем проекте? Различаете ли вы проверку и приёмку?

### Практики описания системы

Практики описания системы — это просто практики записи, они не подразумевают какого-то выдумывания описываемой системы. Практики описания включают в себя

знания по языку описания, используемым нотациям, описательным идиомам, редакторам. Эти практики описания не нужно путать с практиками инженерии (инженерии требований, инженерии системной архитектуры, проверки и приёмки). Практики инженерии учат тому, как придумать соответствующие определения системы (альфы). А практики описания — как создать рабочие продукты, документирующие принятые при определении системы инженерные решения. То, что вы знаете, как поименовать какой-нибудь модуль и как его описать в разных видах моделей, используемых в инженерном проекте, ещё не означает, что вы можете этот модуль для системы предложить и правильно определить его интерфейсы, минимизировав связи с другими модулями. Умение нарисовать правильным значком компоненту на принципиальной схеме вовсе не означает, что вы понимаете, как создать эту принципиальную схему так, чтобы результирующая система была успешной.

Писец (писарь), пишущий под чужую диктовку знакомыми ему иероглифами — это не инженер. Это писец. Инженер — это тот, кто придумывает инженерные решения, которые потом (обычно сам!) записывает иероглифами соответствующих нотаций, плюс потом инженер ещё и воплощает придуманное.

## *Требования*

### Два смысла слова “требования”.

Термин “требования” имеет два разных смысла:

- Определение системы как “чёрного ящика” — что требуется от целевой системы с точки зрения её стейкхолдеров (stakeholder requirements) и использующей надсистемы (system requirements). Обычно это спецификация (т.е. точное формулирование) способности (capability) или условия (condition), которое должно или может быть удовлетворено (функция, которую нужно будет выполнить, или характеристика, которую нужно достигнуть и т.д.).
- Любые определения системы (”чёрного ящика”, “прозрачного ящика”, на любых стадиях жизненного цикла), в которых присутствует деонтическая модальность (модальность долженствования).

Так что лучше не использовать слово “требования” (ибо непонятно, о чём идёт речь), а использовать уточняющие определения: в системной инженерии принято говорить о требованиях стейкхолдеров и системных требованиях, а всякие остальные “требования” (требования стандартов, требования системной архитектуры, требования чертежей, требования проектной документации и т.д.) просто означают, что “система должна удовлетворить этим описаниям”, но это не будет “требованиями” в смысле системной инженерии.

“Требования” обычно означают совокупность отдельных “требований” (отдельных высказываний о системе).

### Модальности в требованиях

Чтобы понять природу требований, нужно разобраться с логическими модальностями высказываний о системе (http://en.wikipedia.org/wiki/Modal_logic):

- нейтральные высказывания о мире, суть которых непонятна без указания модальности. Например, "длина дана как 16".

- алетическая (alethic) модальность, относящаяся к возможности существования. Обычно с алетической модальностью в связи 4D extensionalism связаны рассуждения про возможные миры (possible worlds): пока воплощения системы ещё нет, возможны варианты определений системы для разных возможных вариантов будущего воплощения системы — но только один из них будет реализован "в железе". Например, "длина пусть будет 16" ("положим длину равную 16").
- деонтическая (deontic) модальность, относящаяся к обязыванию и дозволению. Например, "длина должна быть 16". Собственно, это и есть главная модальность "требований", требования — это то, что должно быть, рекомендуется быть, разрешается быть, обязательно или необязательно и т.д.
- темпоральная (temporal) модальность, связанная со временем. Например, "длина была 16 три года назад".
- Доксическая (doxastic) модальность, связанная с верой. "Я верю, что длина дана как 16". Доксическая модальность важна для квалификации (удостоверения того, что требования выполнены — вера в то, что система соответствует её определению).

Требования довольно трудно формализовать именно потому, что нужно разбираться с их модальностями. Сложность вопроса можно оценить по базовой статье, только не в попсовой википедии, а в серьезной философской энциклопедии: http://plato.stanford.edu/entries/logic-modal/

### Инженерные обоснования

Требования связаны с инженерными обоснованиями: они переформулируются как "декларации" (claim) разработчиков о соответствии — т.е. "я верю, что длина равна 16", а затем это высказывание доказывается по логическим правилам рациональной аргументации (помним, что логика — это дисциплина, занимающаяся правильностью рассуждений). Эти доказательства проводятся "как в суде" — и для этого даже заводится "дело" (assurance case, как раз от "судебного дела" — с вариантами dependability case, safety case, security case, requirement case, architectural quality case).

Обзор по инженерным обоснованиям приведён тут: http://ailev.livejournal.com/811715.html

### Рабочие продукты требований

Рабочие продукты требований могут быть самые разные — и чаще всего они не называются "требования". Так, требования можно обнаружить в:

- Разделе "технические требования" технических заданий (хотя "техническое задание" чаще всего подробно перечисляет работы — "задание на работы", а не требования, но всё-таки какое-то описание целевой системы как "чёрного ящика" это описание содержит)
- Документе "Опросный лист" (который посылается, чтобы опросить потенциальных поставщиков инженерных решений и содержит как раз основные требования к поставляемой системе) и посылаемом в ответ на "Опросный лист" документе "Технико-коммерческие предложения"
- Различных стандартах (некоторые из которых называются "технические

условия”, т.е. даже в названиях они не содержат слова “стандарт” или “требования”)

Важно понимать, что требования системной инженерии — это какие-то определения “чёрного ящика”, которые могут даже не содержать слово “требования” в своих рабочих продуктах (могут использоваться слова “спецификация”, “стандарт”, “опросный лист”, “тактико-технические данные/характеристики” и т.д.). А требования в общем смысле слова могут и не содержать слов, обозначающих деонтическую модальность — быть без слов “должно”, “возможно”, “обязательно” и т.д. (это слово может быть, например, использовано в отдельном рабочем продукте — приказе, который делает какой-то документ с описанием системы дОлжным к соблюдению).

Требования совсем необязательно являются бумажными документами-текстами. Они вполне могут храниться в какой-то базе данных (часто эта база данных ведётся в рамках какой-то “информационной системы управления требованиями” — DOORS, IRqA и т.д.) в виде отдельных пронумерованных текстовых описаний. Или требования могут быть представлены в виде какой-то компьютерной модели, численной или логической.

### Требования стейкхолдеров

Требования стейкхолдеров — это требования (описания “черного ящика”), которые создаются для отдельных стейкхолдеров. Конечно, требования разных стейкхолдеров будут противоречить друг другу. Обычно от инженера по требованиям требуется документировать (в тексте или какой-то модели) требования стейкхолдеров, а затем завизировать эти требования у них — чтобы подтвердить правильность понимания. К требованиям стейкхолдерам главное — их понятность самим стейкхолдерам.

Нужно понимать, что стейкхолдеры имеют свои нужды (needs), ибо каждый из них участвует в какой-то деятельности, которая диктует свои цели. И они нуждаются в требованиях, которые сфокусированы их нуждами: если клиенту нужна связь, бесполезно писать требования на обувь или холодильник. Нужно будет писать требования на телефон, или радиостанцию, или лазерную линию связи — желательно при этом, чтобы поначалу требования на связь были абстрагированы от конкретного варианта связных устройств.

### Требования и ограничения

Конечно, каждый клиент мнит себя инженером (а иногда не мнит, а действительно является инженером, а иногда ещё и инженером, более знающим, чем инженеры команды проекта). Такой клиент не только будет формулировать требования стейкхолдера, а также требования к системе, но обязательно попытается сформулировать конкретные инженерные решения “прозрачного ящика” (например, из каких частей должна составлять система, какое в ней должно быть использовано оборудование). Формально высказывания о “прозрачном ящике” не называются требованиями, поэтому некоторые авторы предлагают называть их “ограничениями” (свободы творчества инженерной команды).

Неплохо бы понимать в каждом проекте, что является требованиями, а что является ограничениями. Прежде всего вы должны удовлетворить требования. И если придуманное вами инженерное решение для “прозрачного ящика” лучше того, которое требует клиент в своих ограничениях, нужно попытаться убедить клиента снять эти ограничения.

Но нужно также понимать, что бывают ситуации, когда эти ограничения отражают какой-то опыт клиента, неизвестный команде, или они появляются из неинженерных (политических, финансовых, логистических и т.д.) соображений. Поэтому по поводу ограничений нужно каждый раз понимать, почему они были прописаны, почему клиент без них не может обойтись.

### Требования к системе

Требования стейкхолдеров могут быть противоречивы, разрознены и неполны. Требования, достаточные для разработки системы, называются системными требованиями (требованиями к системе, system requirements). Их разрабатывает инженер по требованиям в ходе так называемой "аналитической" работы (хотя в этой работе кроме анализа требований стейкхолдер и присутствует синтез системных требований).

К требованиям к системе предъявляют множество самых разных требований: непротиворечивости, полноты, проверяемости и т.д. Разработать хорошие требования к системе очень и очень непросто.

Главная ошибка, из-за которой терпят провал многие проекты — это необнаружение какого-то важного требования какого-то из стейкхолдеров.

### Инженерия требований

Разработкой требований занимается поддисциплина системной инженерии "инженерия требований". Главная часть инженерии требований — это реверс-инжиниринг использующей (над)системы (using system) для того, чтобы получить описания "чёрного ящика". Инженеры по требованиям (это вид системных инженеров) выявляют стейкхолдеров, узнают их потребности (нужды, needs), разрабатывают требования стейкхолдеров, проводят переговоры между стейкхолдерами в тех случаях, когда их требования противоречивы, или когда команда проекта оказывается не в состоянии их выполнить и требования нуждаются в коррекции, согласуют требования стейкхолдеров между собой, готовят требования к системе.

Неправильно называть инженерию требований "управлением требованиями". Управление требованиями — это дисциплина инженерного менеджмента (логистическая, "инженерного документооборота"), она заключается в том, чтобы предоставить средства хранения требований и сообщения о них всем тем, кто в них нуждается. Для того, чтобы управлять требованиями, нужно их сначала иметь: а для этого нужно проводить инженерную работу (прежде всего делать реверс-инжиниринг использующей системы, чтобы понять, что в ней должна делать целевая система). Обычно управление требованиями включают в состав инженерии требований как поддисциплину, но только одну из многих. Обзор стандартов представления требований в целях управления требованиями (хранения и пересылки) можно найти тут: http://ailev.livejournal.com/900086.html — обратите внимание, что во всех этих стандартах нет никаких следов того, откуда эти требования берутся и как потом используются.

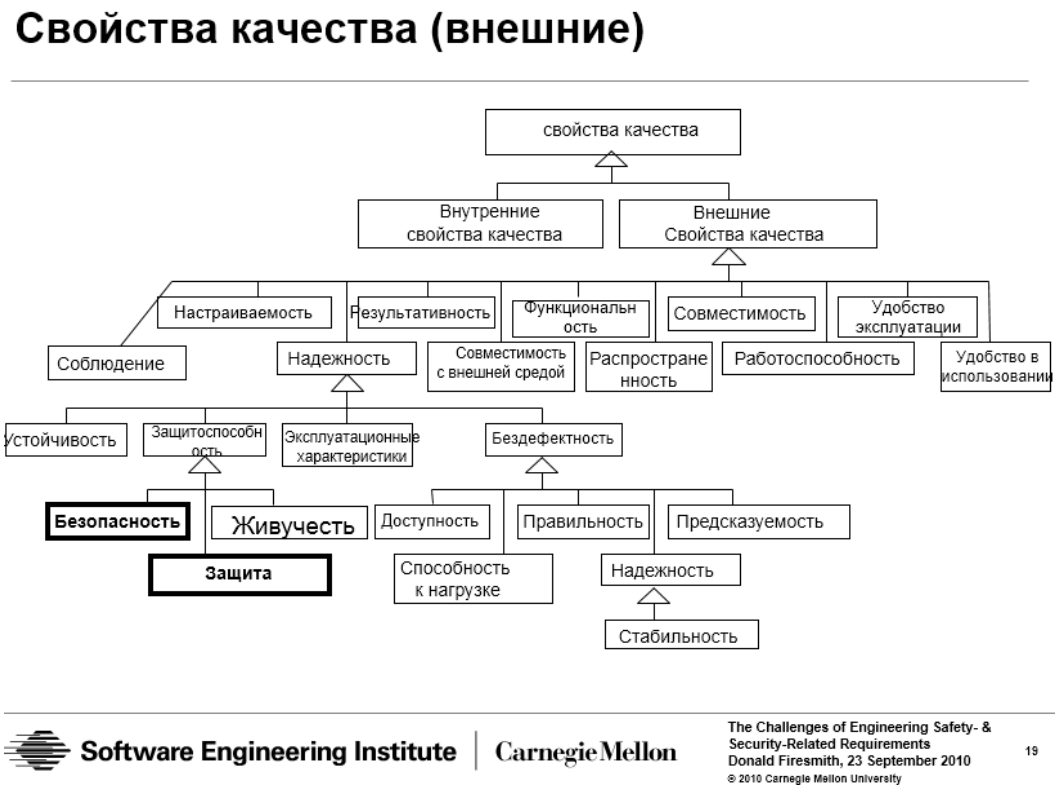
Инженеры по требованиям получают подготовку, которая немного отличается от подготовки инженеров-архитекторов: им нужно в бОльшей степени владеть навыками коммуникации, ибо они должны постоянно общаться с самыми разными стейкхолдерами не-инженерами. Также им полезно знать азы конфликтологии (чтобы улаживать противоречия между стейкхолдерами), и понимать основы инженерного менеджмента (ибо им тесно приходится работать с альфой

“возможности” — именно “возможности” фокусируют требования).

### Какие бывают виды требований

Самые распространённые практики инженерии требований — это выявление функций (поведения) системы из каких-то сценариев взаимодействия (user stories, use cases: там множество вариантов). Иногда про такие требования, выведенные из сценариев использования, говорят “функциональные требования”, противопоставляя их “нефункциональным” (например, требованиям надёжности, ремонтопригодности, доступности, безопасности и т.д., так называемые “-ости”, по-английски это будут “ilities” — reliability, repairability, availability, safety, etc.). Но есть замечание Donald Firesmith, что “не бывает нефункциональных требований” — ибо все эти “требования качества” (как их иногда тоже называют) это абсолютно функциональные требования, характеризующие функции системы с точки зрения каких-то стейкхолдеров, обычно не рассматривающихся в сценариях “пользования”.

Видов требований существуют десятки, но принадлежность к этим видам не так уж важна: если вы встретили в пустыне льва, вам же не нужно знать, что он из семейства кошачьих? Много важнее заметить, что этот лев рядом с вами! Главный источник ошибок в проекте — это неведение относительно наличия каких-то требований. Впрочем, классификация может помочь, если вы зададите себе вопрос: какие виды требований вы ещё не рассматривали для вашего проекта? Вот пример классификации требований качества — знаете ли вы их для вашего проекта?




Подробнее про требования защитоспособности (выделенные на рисунке выше) можно посмотреть в презентации Дональда Файерсмита — http://vniiaes.ru/HTML/RU/Docs/RuSEC%202010.rar (и там же можно посмотреть на презентацию по целеориентированной инженерии требований Яна Александера).

### Кто должен делать требования

Иногда между командой инженеров и заказчиком можно услышать следующий разговор со стороны инженеров: “вы предоставьте нам подробные требования,

иначе мы и пальцем не пошевелим". Должен разочаровать: это неверная позиция. Заказчик может представить только свои нужды, а не требования к системе. Если у заказчика есть инженеры (или заказ на разработку системы делает инженерная фирма), то есть шанс получить "требования стейкхолдера". Но это ещё не требования к системе. Разрабатывает требования обычно команда проекта, они не приходят извне проекта. Другое дело, что без активного участия стейкхолдеров (в том числе представителей организации заказчика) требования не подготовишь. Но в любом случае, ответственность за требования лежит на команде проекта, а не на заказчике.

Заказчик же может, например, завизировать разработанные командой проекта требования — но и в этом случае, изготовленная система, которая не удовлетворит его нуждам (needs) по причине того, что требования были сформулированы командой проекта неправильно, может встретить серьёзные проблемы при приёмке/validation (хотя блестяще пройдёт проверку/verification — инженеры подтвердят сами себе, что изготовили ровно то, что планировали, хотя это не то, в чём нуждался клиент).

Почему за требования ответственен инженер? Потому что он должен сделать реверс-инжиниринг использующей системы: часто клиент на это не способен, хотя он обычно и весьма знающ об особенностях используемой системы, но эти знания могут быть совсем не такими, на которые мог бы расчитывать инженер.

### Целеориентированная инженерия требований

Целеориентированная системная инженерия (goal-oriented requirements engineering, GORE) подразумевает более или менее формальное описание дополнительных по отношению к требованиям сущностей (таких как стейкхолдеры, цели, влияния, аргументы в защиту, альтернативы и т.д.). Это даёт возможноть более точно документировать происхождение требований, обосновывать выбор требований, связывать выбор проектных решений (архитектурных решений) с требованиями. Получаемые сложные структуры всех этих стейкхолдеров-целей-влияний-альтернатив-требований-аргументов-и_т.д. называют моделями требований, а использующую такие модели инженерию требований – моделеориентированной инженерией требований. В инженерии предприятий (enterprise engineering) модели требований [к предприятию/к архитектуре предприятия] часто называют мотивационными моделями (motivation не только «движущая сила, побуждение», но и обоснование – reason, justification).

Примерами языков для таких моделей требований могут быть GRL (http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome), подходе i* (этот подход родоночальник и классика GORE — istar.rwth-aachen.de), Motivational Extension в ArchiMate 2.0 (http://pubs.opengroup.org/architecture/archimate2-doc/chap10.html#_Toc371945252), Model-based requirement discovery от Яна Александера (http://www.scenarioplus.org.uk/papers/mbrd/Model-based_Requirements_Discovery.PDF).

Моделеориентированность (как шаг к формальным требованиям) хороша для проведения проверок самих требований (хотя она не поможет найти пропущенное требование) – смотри вопросы Яна Александера по валидации требований на основе модели требований (слайд 26 по предыдущей ссылке).

Требования должны документироваться и затем использоваться для проведения испытаний (проверок и приёмок) как чеклисты. Проверки – это насколько

правильно изготовлен «чёрный ящик», приёмки – это насколько использование целевой системы позволяет пользователю добиться целевого поведения от using system. Test driven development – это когда разработка системы начинается с того, что требования разрабатываются сразу в виде тестов (исполняемых!). Увы, качество разработки по TDD получается примерно такое же, как и при любых других видах разработки – что не снижает популярности этого подхода.

Требования как правило разрабатываются с помощью сценариев (use cases), в которых рассказывается, что делает система в ответ на последовательности действий (сценарии) с участием разных стейкхолдеров. Более простая форма – user story, в которых рассказывается, что делают стейкхолдеры с системой (а не что делает система!).

Самая простая форма моделеориентированности – это использовать user story в формулировках по шаблону (Mike Cohn, 2008, Advantages of the "As a user, I want" user story template, blog post, http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template): Я как __стейкхолдер__ хочу, чтобы система ___формулировка требования___, для того чтобы ___хотелка-для-using-system___.

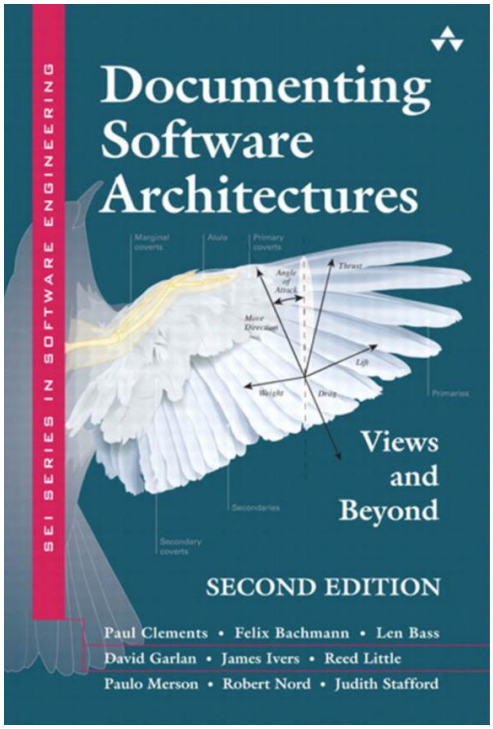
*Архитектура*

Системная архитектура — эта альфа, пожалуй, важнейшая среди подальф определения системы. Она была сформулирована по мотивам строительной архитектуры относительно недавно (лет эдак двадцать назад) и имеет множество определений:

- Из ISO 42010: Архитектура (системы) – основные понятия или свойства системы в её среде, заключающиеся в её элементах, их отношениях и принципах её проектирования и развития (Architecture (of a system) – fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution).
- Из книги Garlan et al.: Архитектура системы это набор структур, необходимых для рассуждений о системе, каковые структуры состоят из элементов, отношений и свойств этих элементов и отношений (The architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both).
- Набор из более чем 150 других определений архитектуры, которые предлагались профессионалами системной и программной инженерии: http://www.sei.cmu.edu/architecture/start/glossary/community.cfm

После одной из затяжных интернет-дискуссий по архитектуре в программной инженерии Ralf Johnson выдал альтернативное определение: "Архитектура — это обо всём важном. Что бы это ни было". А что можно считать важным в инженерной системе? Если представить архитектуру как набор инженерных решений, принимаемых для определения структуры системы, то "важные решения" — это при изменении которых приходится в существенной мере изменять множество других решений, в существенной мере перепроектировать систему. Если вместо двигателя внутреннего сгорания на автомобиль поставить электромоторы, то автомобиль придётся перепроектировать практически заново (хотя что-то из неархитектурных решений может и остаться: оплётка руля, стекло на дверцах, фары). Следовательно, выбор между двигателем внутреннего сгорания и электродвигателем в ходе создания автомобиля — это архитектурное решение.

Книга Paul Clements, et al. “Documenting Software Architectures” упоминается в литературе к стандарту архитектурных описаний ISO 42010. Она предлагает метафору для понимания того, что такое архитектурное описание и почему его так трудно сделать: как документировать крыло птицы для того, кто не знает, что это такое? Там ведь огромное количество разноуровневых структур, повторения с вариациями, критические для полёта качества элементов и свойства крыла, невыводимые из свойств отдельных его частей. Как записать все эти структуры, в которых каждое перо неповторяемо в его даже самых маленьких деталях, есть множество особенностей костей и мышц, но все они в совокупности дают функцию полёта?




Архитектура — это основные описания “прозрачного ящика”, которые показывают, из каких основных частей (компонент, модулей, размещений и т.д.) состоит система, и как эти описания “прозрачного ящика” обеспечивают функцию “чёрного ящика”.

Итого: важно, какие типы структур системы документируются (например — структуры компонент/принципиальные схемы, модули и их интерфейсы, размещения и т.д.), какие принципы проектирования/конструирования и развития документируются.

Кем и для чего используются архитектурные описания, подготавливаемые практикой инженерии системной архитектуры? Прежде всего — командой проекта для обсуждений системы. Архитектурные описания для инженеров как шахматная доска для игроков в шахматы — это гроссмейстеры могут играть “в уме”, а простые люди предпочитают вести сложные рассуждения не в кортексе (коре головного мозга), а с использованием экзокортекса, тем более при командной работе. Команда поможет также убрать ошибки: эти ошибки трудно заметить, если обсуждаемые важные инженерные решения не задокументированы, если отсутствуют архитектурные описания, а решения только проговариваются вслух (а иногда ведь решения даже не проговариваются — их принимает кто-то один, а другие о принятии этих решений и их важности и не догадываются!).

Особенно полезна архитектура для рассуждений по «требованиям качества» («-ости»: надёжность, ремонтопригодность, безопасность и т.д.). Именно в ходе

разработки архитектуры увязываются обсуждение виброшумовой характеристики, энергоэффективности, долговечности и надёжности, материалоёмкости.

Архитектурные инженерные решения ясно выражают ограничения для для проектантов/конструкторов (команды проекта), поэтому они стабилизируют разработку.

Архитектурные описания также помогают разъяснять принимаемые по поводу целевой системы инженерные решения для внешних по отношению к команде стейкхолдеров («шахматная доска для зрителей и судей»).

Архитектурную работу можно делать плохо, а можно хорошо (но делается она в любом случае). Чтобы её делать хорошо, нужно её обсуждать специально, знать лучшие архитектурные практики!

**Инженерия системной архитектуры**

Практика создания системной архитектуры получила название "инженерия системной архитектуры". Конечно, у этой практики есть и множество других названий: architecturing (архитектурирование — но так по-русски говорят редко), эскизное проектирование (conceptual design). Инженерия системной архитектуры представляет собой часть в архитектурном проектировании (architecturing design), хотя в это проектирование входит создание всего проекта (design), в том числе и неархитектурной его части.

Создание системной архитектуры подразумевает синтез самых разных описаний целевой системы и прямой инженерный творческий процесс (в отличие от реверс-инжиниринга, т.е. "обратной инженерии" использующей системы в инженерии требований. Реверс-инжиниринг считается главным образом аналитической практикой — т.е. практикой не придумывания, а выявления/discovery описаний, в чём-то сродного научной работе, а не инженерной).

Как и любое определение системы, архитектура у системы есть всегда, но не всегда при разработке тщательно делаются архитектурные описания. Часто это «устная архитектурная традиция» — и тогда архитектуру называют "заделы", "опыт", "наработки" (обычно при произнесении этих слов как раз и имеются ввиду важные инженерные решения, каковы бы они ни были). Часто архитектурные решения передаются из уст в уста, как «народный эпос».

Более того, раньше при наличии требований и существовании этого "народного эпоса" часто переходили непосредственно к проектированию и конструированию, не документируя принятых важнейших решений — особенно, когда эти решения принимались методом "проб и ошибок". Архитектурное знание не накапливалось, и не обсуждалось, ибо не было документировано. Сейчас же принято документирование архитектурного знания, поэтому "проектирование" всё чаще называют "архитектурным проектированием" (например, в стандарте ISO 15288 практик жизненного цикла системной инженерии нет практики "проектирования", а есть именно практика "архитектурного проектирования").


Системная архитектура разрабатывается системными архитекторами, специализация на системной архитектуре одна из основных специализаций для системных инженеров. Часто "системный инженер-архитектор" сокращают до просто "архитектор", но нужно учитывать возможную путаницу со строительными архитекторами.

Ещё одно название для архитектурного проектирования — системное проектирование, ибо само понятие архитектуры с его множественностью структур системы и зависимости определения этих структур от нужд и интересов стейкхолдеров опирается на системный подход.

### Практики архитектурного проектирования

Их множество (см. обзор в первой главе книги М.Левина «Технология поддержки решений для модульных систем»: http://www.mslevin.iitp.ru/Levin-bk-Nov2013-071.pdf). У всех них один «недостаток»: хорошему инженеру эти методы помогают, плохому инженеру они помочь не могут.

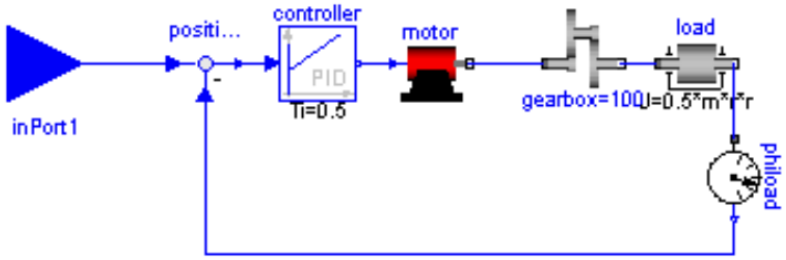
Наиболее известен ТРИЗ (http://ru.wikibooks.org/wiki/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B_%D0%A2%D0%A0%D0%98%D0%97, можно начинать с проглядывания трёх текстов: АРИЗ-85В http://www.triz-ri.ru/triz/triz02.asp, типовых приёмов разрешения противоречий http://www.triz-ri.ru/triz/triz04.asp и стандартных решений изобретательских задач) как метод совмещения логической и физической архитектур.

DSM (http://www.dsmweb.org/) чаще всего используется как метод разбиения системы на модули.

ТРИЗ и DSM практикуются во многих центрах разработки. Но есть огромное число методов, практикующихся в рамках одного-двух университетских или промышленных исследовательских центров/лабораторий (реже – центров разработки).

Все более и более распространены архитектурные расчёты, в которых принципиальная схема "живая" и по ней можно вести мультифизические расчёты. Вот пример такой архитектурной модели привода на акаузальном языке мультифизического моделирования Modelica (https://modelica.org/ — там тонны материала, разберитесь! Бесплатный софт для Modelica —

https://openmodelica.org/):




На сегодня активно развиваются компьютерные методы не только «проверочных архитектурных расчётов», но и методы оптимизации (выбора оптимальной архитектуры, т.е. оптимального набора компонент и реализующих их модулей).

Из языков архитектуры предприятия рекомендуется ArchiMate 2.0: стандарт — http://pubs.opengroup.org/architecture/archimate2-doc/, свободный моделер — http://www.archimatetool.com/, лучшая книжка вот тут: http://masteringarchimate.com/mastering-archimate-edition-ii/. Есть русский перевод терминологии: http://ailev.livejournal.com/988360.html

Тут нужно бы обсудить и product lines (например, подход для software product line practice, версия 5 — http://www.sei.cmu.edu/productlines/frame_report/what.is.a.PL.htm), и светлое будущее (поиск-ориентированная инженерия и т.д. — http://ailev.livejournal.com/1122876.html), но они для нашей книги факультативны.

### Минимальная архитектура

Минимальная архитектура состоит из трёх определений (и, соответственно, трёх тематических описаний — наборов рабочих продуктов/моделей):

- структуры компонент и соединений (чаще всего — принципиальной схемы, иногда просто функциональной декомпозиции). Эту часть архитектуры также называют “логической архитектурой”.
- Структуры модулей (определяемых их интерфейсами, а часто и физической декомпозицией). Эту часть архитектуры называют также “физической архитектурой”.
- Указаний на размещение частей системы в пространстве.

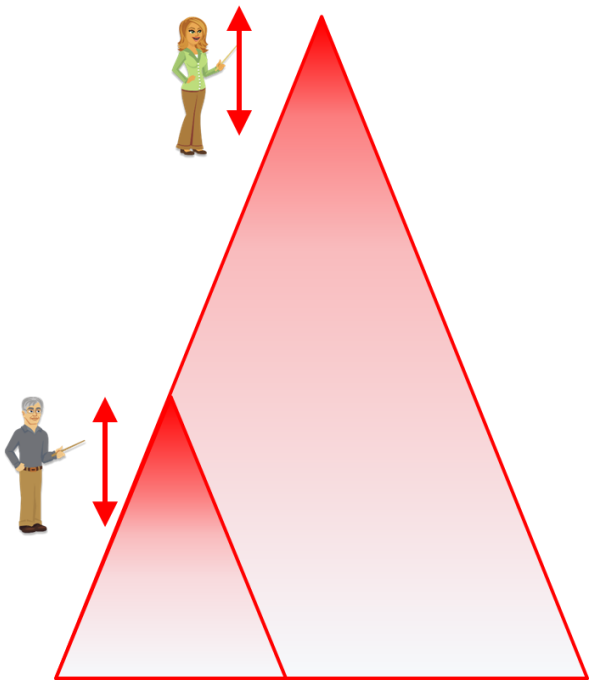
Обратим внимание, что архитектура должна чётко показывать, как увязанные между собой все эти три вида описаний (иногда говорят о “стилях описаний”, а не “видах описаний”) “прозрачного ящика” дают на выходе основные функции системы. Тем самым архитектура неразрывно связана с требованиями. Те требования, которые максимально влияют на архитектуру, называют архитектурными требованиями.

Чаще всего разработка начинается с предложения принципиальной схемы, или функциональной декомпозиции (“логической” схемы) и проведения необходимых мультифизических расчётов (в том числе с учётом киберфизической составляющей: часть физических характеристик может быть обеспечена активным компьютерным управлением). Потом делается попытка определить то, как компоненты будут реализованы теми или иными модулями — которые можно купить, или которые придётся разработать. Часто выясняется, что предполагаемые принципиальной схемой модули трудно реализовать (их нет на рынке, их трудно спроектировать и изготовить и т.д.). Тогда принципиальная схема меняется, и делается очередная

попытка разобраться с модулями оборудования и деталями. И так до тех пор, пока логическая и физическая архитектуры не окажутся согласованы между собой.

Субъективность и относительность архитектуры.

По поводу архитектуры субъективность проявляется во многих аспектах. Так, архитектура у системы есть всегда — но если попросить для уже готовой системы разных инженеров сделать архитектурные описания, то они будут разными — как по предложенным структурам системы, так и по той границе, которую проводят разные инженеры между "важным" и "неважным", исходя из своего опыта. Если предложить разным системным архитекторам спроектировать какую-то систему, то разные системные архитекторы также предложат разные архитектуры — но в данном случае они их "придумают", а не "выявят". Более того, один и тот же системный архитектор (или команда системных архитекторов) не только могут, но и должны предложить сразу несколько разных архитектурных решений (вариантов компонентной структуры, выбор разных модулей для реализации компонент, разного размещения в пространстве). Лучшие из этих архитектурных решений выбираются при помощи процедуры, известной как trade-off studies (перевод по-русски тут немного нетрадиционен: "прохождение развилок") — часто готовятся таблицы, в которых баллами оцениваются свойства альтернативных вариантов, а затем подсчитывается сумма с учётом каких-то коэффициентов. В реальной жизни решения обычно принимаются содержательным обсуждением, а затем таблицы для trade-off studies оформляются задним числом для отчётности и демонстрации наличия вариантов, учтённых в работе — это как бы "объективация" субъективно принимаемых решений. Если рассматривалась только одна архитектура, то это считается не слишком хорошей работой системного архитектора.



Субъективность архитектурных решений проявляется и в том, что где заканчиваются архитектурные (важные) решения, а где начинается конструкторская и проектировочная рутина, не затрагивающая систему в целом знает только системный архитектор — никаких на этот счёт рекомендаций, стандартов, учебников нет. На диаграмме наверху это показано как градиентная размывка решений: число переделок, требуемых при изменении решений плавно

уменьшается так, что чёткой границей архитектурной и неархитекторной части проекта провести нельзя.

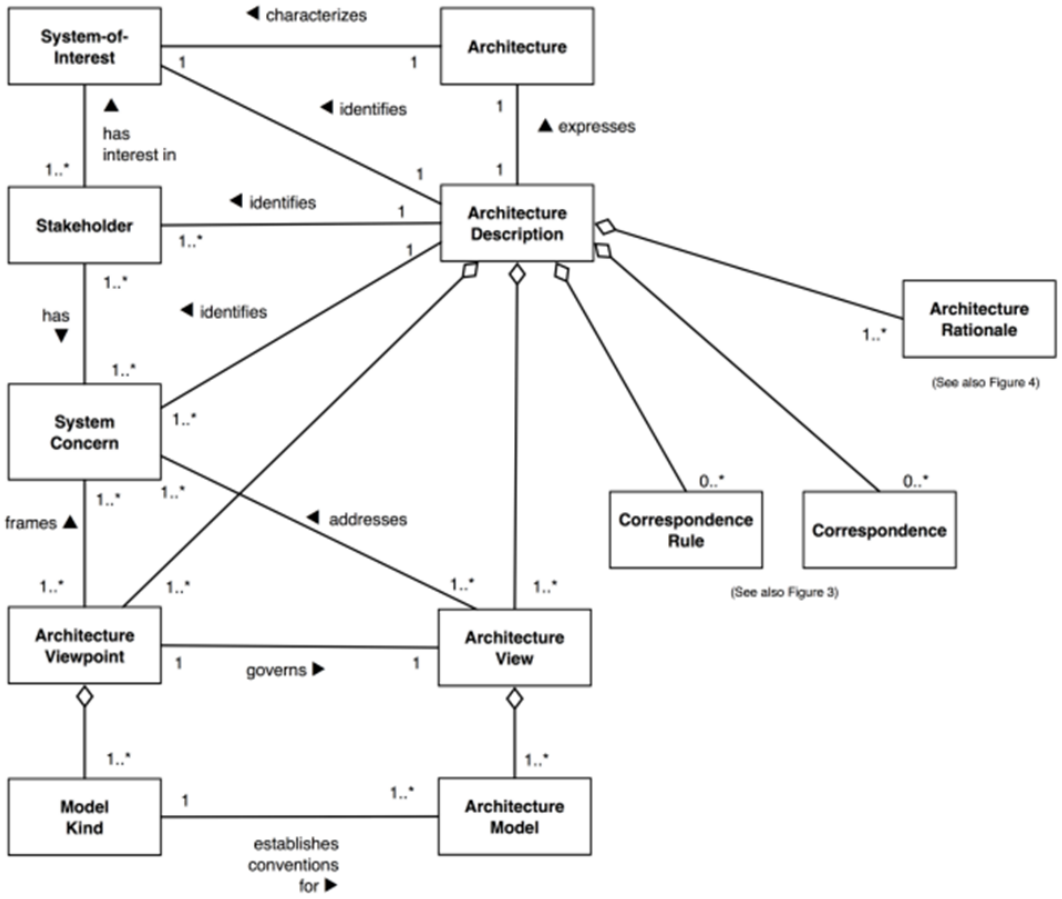
Относительность архитектурных решений заключается в том, что "неархитектурные решения" для системного архитектора надсистемы могут быть "архитектурными решениями" системного архитектора подсистемы (для этого архитектора это ведь будет не "подсистема", а целевая система). Так, архитектура авиадвигателя является таковой только для архитектора авиадвигателя. Для архитектора самолёта все решения внутри двигателя (например, как устроены форсунки реактивного двигателя) не архитектурны, архитектурен только выбор модуля двигателя — если выберется не реактивный двигатель, а поршневой, то придётся менять довольно много конструкторских решений для всего самолёта.

Так что критерий, где остановиться системному инженеру, спускаясь по отношениям часть-целое в структуре системы прост: останавливайтесь там, где вы

- дошли до того уровня деления системы на элементы, на котором вам кажется, что уже нет важных *ваших* решений.
- уже поделили работу между отдельными исполнителями-разработчиками модулей и дальше будут *их* важные решения.

Архитектурные описания

Архитектура выражается в архитектурных описаниях (рабочих продуктах). Вот диаграмма из ISO 42010, демонстрирующая связь архитектуры и архитектурных описаний:




Замечания по переводу: очень часто view и viewpoint переводят как "аспект", но это не совсем верный перевод (тем более, что он не позволяет различить само

описание, выполненное для какого-то аспекта — view, и знания о том, как делать такие описания — viewpoint). View — это то, что видишь. Viewpoint — это та точка, из которой видно то, что видишь. Видишь обычно много разного (разных моделей), объединённого тематикой: описания требований, архитектуры, финансовые описания, физические описания, описания безопасности и т.д.

### Как объединять разные модели и группы описаний

Correspondence rule — это правила соответствия элементов в разных моделях, они определяют конкретные соответствия (например, правила соответствия компонент и модулей в инженерной группе описаний. Эти правила определяют набор конкретных соответствий компонент и модулей в данном проекте).

Всё, что говорится про архитектуру и архитектурные описания, верно для двух подходов к конструированию групп описаний из отдельных моделей (диаграмм, наборов формул и т.д.):

- Синтетический подход — когда отдельные модели (например, диаграммы) являются именно отдельными моделями, но к ним добавляется список соответствующих друг другу элементов (”насос-1 на принципиальной схеме реализуется модулем “наклонный жёлоб” на чертеже”).
- Прожекторный (проекторный) подход — когда вся необходимая информация разных моделей содержится в общей базе данных (иногда говорят “интеллектуальной информационной модели”, “цифровом макете”, “цифровой модели” и т.д.), а отдельные модели получаются отфильтровыванием нужной информации из общего хранилища — примерно так, как из белого цвета в проекторах и прожекторах получают нужные цвета, используя цветные фильтры-плёнки.

Эти два подхода логически эквивалентны.

### Архитектурные модели и другие виды описаний

Современный тренд в архитектурных описаниях — это использование формальных (понимаемых компьютером) моделей и задание формализмов в качестве метода описания. Некоторые авторы пытаются даже сказать, что если это не архитектурные модели, то диаграммы и тексты не должны входить в состав архитектурного описания. Но это не так: в архитектурных описаниях, конечно, могут присутствовать и не-модели, например:

- Архитектурные текстовые эссе оказываются очень полезными для краткого описания того, что было положено в основу моделей архитектурных описаний (выбор метода моделирования), краткого обзора архитектуры в целом как совокупности моделей. Эти же эссе могут быть использованы для пояснений того, как именно были связаны архитектурные модели, а также особенностей моделирования.
- Инфографика (а хоть и слайды в PowerPoint) может показать основные архитектурные идеи для не-инженеров (менеджеров, заказчиков, пользователей) — ибо формальные модели могут быть для них непонятны.

В архитектурные описания обязательно входят ещё и архитектурные обоснования (резоны, по которым были приняты те или иные архитектурные решения — architecture rationale).

Тем самым архитектурные описания — это документированные (в том числе в

компьютерных формах документирования — моделях, базах данных, файлах САПР) важные инженерные решения, а не:

- «архитектурная документация» (документация часто только выписка из компьютерной модели)
- «несколько главных документов проекта» (а столько, сколько нужно),
- «несколько первых уровней декомпозиции» (а столько, сколько нужно),
- «отсутствие детальной проработки (деталей будет столько, сколько нужно).

Архитектурные знания

Очень редко архитектурные (важные, требующие существенных переделок всего проекта при их изменении) решения изобретаются заново, чаще они повторноиспользуемы. Знания — это повторноиспользуемая в разных ситуациях информация, т.е. можно говорить об архитектурных знаниях.

Архитектурные решения – это чаще всего накопленные человечеством, отраслью, организацией, конструктором/проектировщиком знания. Чаще всего архитектура не "изобретается", а просто дорабатывается уже готовая — якобы "новая система" использует огромное количество старых важных инженерных решений.

Тем не менее лидерство в инженерии обычно определяется предложением новых архитектур — новых инженерных решений, т.е. связано с развитием инженерного знания, обычно исходя из "первых принципов" (т.е. исходя из физических законов, теории алгоритмов и т.д.).

### *Неархитектурная часть проекта*

Неархитектурная часть проекта обычно называется "рабочим проектом", а описания — "рабочей документацией" (хотя есть нюансы: в рабочий проект может входить и архитектура, а рабочая документация выполняться не в виде документов, а быть каким-нибудь "цифровым макетом").

Рабочая документация включает в себя технологическую часть: информацию о том, как изготовить и собрать систему с использованием конкретного оборудования. Поэтому рабочую документацию часто готовят конструкторские и проектные бюро при заводах и строительно-инженерных компаниях: они хорошо знают особенности своих технологий и используют их при проектировании.

В любом случае риск "напортачить" при принятии инженерных решений в рабочем проекте невелик: конечно, некачественно спроектированная какая-нибудь скобка на корпусе может привести к поломке целевой системы, сломавшись в реальной системе. Но перепроектирование её (принятие альтернативного инженерного решения) не приведёт к перепроектированию большого числа других элементов системы — поэтому рабочее проектирование/конструирование делают уже не системные инженеры.

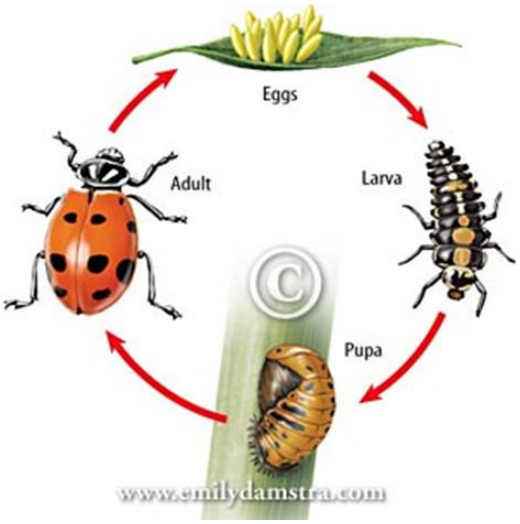
Есть множество самых разных описаний, которые входят в рабочую документацию — не только чертежи и сборочные диаграммы, но и программы (планы) испытаний, инструкции по эксплуатации, графики проведения работ и т.д.

# 8. Жизненный цикл системы и проекта

## *Понятие жизненного цикла*

### Жизненный цикл чего?

Жизненный цикл (life cycle) — это не "жизненный" и не "цикл". Название произошло от биологического "жизненного цикла", ибо особь каждого вида рождается, живёт, затем умирает. И потом следующая особь рождается — и так далее, по циклу, игнорируя факт, что "по циклу" проходят совсем разные существа.




Технические системы тоже "рождаются" и "умирают", а затем появляются новые системы — но это и не "жизнь" и не такой уж "цикл". Жизненных циклов бывает много разных (системы, проекта, разные виды жизненного цикла), и нужно в этом не запутаться — всех их, конечно, называют просто "жизненный цикл", забывая уточнить, какое использование термина имелось ввиду.

Жизненный цикл системы (system life cycle) — это деятельность всех обеспечивающих систем, ведущих целевую систему от её замысла ("рождения" определения системы) до вывода из эксплуатации ("смерти" воплощения системы), обычно эта деятельность разбита на стадии (stages, в военных стандартах США это фазы, phases), которые вполне могут быть не только последовательными, но и перекрываться во времени друг с другом. Жизненный цикл (деятельность) начинается в какой-то момент времени, а затем заканчивается в какой-то момент времени, стадии его тоже начинаются и заканчиваются в какие-то моменты. Когда говорят "жизненный цикл", то всегда подразумевают полный отрезок времени "от замысла до вывода из эксплуатации", "от рождения до смерти", причём разбитый на стадии. Но отрезками времени не "управляют", а вот когда говорят "управление жизненным циклом" как раз говорят об управлении деятельностью (управлении обеспечивающей системой), обеспечивающей переход от одной стадии жизненного цикла к другой.

Стадии жизненного цикла выделяют по изменению в ходе жизненного цикла преимущественного образа мышления (согласно ISO 24744 — change of mental framework). Это не слишком формальное определение, но оно как минимум не предлагает сосредотачиваться на "состоянии целевой системы", а даётся именно в терминах обеспечивающих систем. На разных стадиях жизненного цикла системы (изделия, установки, сложного инженерного объекта и т.д. — помним, что мы тут про суть дела, а не про терминологию и выбор обозначающих суть дела слов) люди

думают про разное: на стадии проектирования люди думают о проектировании, на стадии строительства о стройке, на стадии эксплуатации — об эксплуатации.

Жизненный цикл проекта (project life cycle) — это часть жизненного цикла системы, которая укладывается в рамки проекта. Иногда жизненный цикл проекта совпадает во времени с какой-то стадией жизненного цикла, иногда не совпадает. Более того, совершенно необязательно, что в рамки жизненного цикла проекта (деятельности проекта) попадает вся деятельность какой-то стадии жизненного цикла системы. Проект обычно бьётся на этапы (чтобы хоть как-то отделять этапы проекта от стадий жизненного цикла).

### Управление жизненным циклом

Управление жизненным циклом (life cycle management) — инженерная дисциплина, в отличие от менеджерской дисциплины управления проектами. Управление жизненным циклом может рассматриваться по-разному:

- Как синоним управления конфигурацией, управления инженерной документацией, управления жизненным циклом продукта (product life cycle management), плюс управление информаций. Основная задача — предотвращение конфигурационных коллизий (т.е. ошибок, возникающих от несоответствия и противоречивости различных документов и моделей друг другу, а также их несоответствие воплощённой системе).
- Как дисциплина, описывающая разные логистические организации распределения инженерных практик (инженерии требований, инженерии системной архитектуры, и т.д.) по стадиям жизненного цикла (последовательное выполнение практик, параллельное выполнение практик, регулярность проведения проверок, ритмичность проведения совещаний и перепланирования, и т.д.).
- Так называется ситуационная инженерия методов с точки зрения менеджеров. Это неслучайно, ибо именно языки и стандарты ситуационной инженерии методов используются для описания жизненного цикла: описываются практики, а затем показывается их распределение по стадиям жизненного цикла. Стандарты ситуационной инженерии методов — OMG Essence, а также часто поминающийся в этой главе ISO 24744.

Проектное управление — это главным образом календарное планирование и контроль выполнения плана, обеспечение плана ресурсами занимает главное место в управлении проектами, отслеживание графика центрально. При этом управление проектами работает именно с проектами, а проекты обычно занимают малую часть жизненного цикла.

Есть особое понимание “управления жизненным циклом” в атомной отрасли, задаваемое документами МАГАТЭ (международного регулятора): у атомщиков управление жизненным циклом понимается как практика продления жизни действующих атомных станций, “управление старением”. Тем не менее, это особое использование термина сегодня сменяется постепенно на общепринятое в системной инженерии понимание.

Управление жизненным циклом в любом случае охватывает полный жизненный цикл системы (т.е. охватывает множество проектов) и сосредотачивается не на “сдаче вовремя”, а на содержательном объединении работ разных стадий жизненного цикла, использовании необходимых инженерных практик. Акцент тут на содержательном change of mental frameworks (изменении преимущественного

мышления) в ходе смены стадий жизненного цикла, а не на точном выполнении графика. В управлении жизненным циклом волнует такая организация работы, которая подразумевает наличие содержательно необходимых ресурсов (а не ресурсов в достаточном количестве для выполнения содержательных инженерных практик, этим занимается проектное управление).

### Типовой жизненный цикл и разнообразие

Типовой жизненный цикл (т.е. деятельность обеспечивающих систем, обеспечивающих именно прохождение целевой системой стадий жизненного цикла) разбит обычно на стадии замысла, проектирования, изготовления, эксплуатации, вывода из эксплуатации. Конечно, в случае разных систем эти стадии могут существенно различаться. Вот пример разнообразия жизненных циклов в части разбиения их на стадии в том виде, как это понимается в ISO 15288:


Это простейший вид: одномерная “колбаска”, стадии которой не перекрываются друг с другом.

### Гейты и вехи

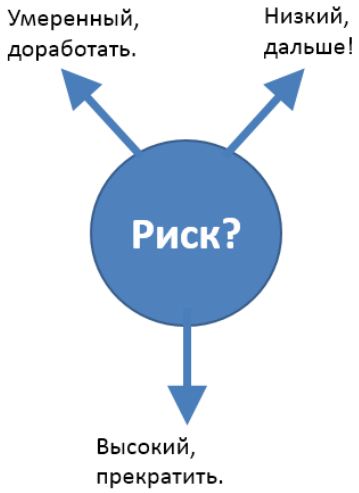
В жизненном цикле переходы со стадию на стадию обычно называют гейтами (decision gate, а перевод “ворота” как-то не прижился, да и слово “решение” тоже как-то потерялось). Гейт обычно находится между стадиями жизненного цикла и связан с окончанием одних проектов (часто выполняемых одними людьми) и началом других проектов (часто выполняемых другими людьми — разные стадии жизненного цикла требуют разной специализации, поэтому состав проекта обычно меняется в ходе разработки). Решение, принимаемое в гейте — это пересмотр выделения ресурсов на проект: синхронизация параллельно ведущихся разработок в инженерной части, а также менеджерской (логистической и инвестиционной) работы.

Вероятность того, что трудности возникнут при стыковке готовых ("в металле", "в бетоне", "в коде" и даже "в голове" для человеко-системной интеграции) частей системы очень велика, поэтому эта стыковка-интеграция и оценка рисков проекта должна проходить не однократно в момент окончания стадии интеграции (изготовления, сборки, наладки) и начала стадии эксплуатации, а существенно чаще, для чего предусматривается несколько таких гейтов — пересмотров выделения ресурсов.

Гейты характеризуются:

- Усиленным контролем конфигурации в эти моменты времени (проходят сверки информационных систем, создание проектных базисов — baselines, т.е. таких "утверждённых версий" проектных моделей и документации, в каких изменения могут затем проводиться только по специальной процедуре со множеством дополнительных проверок),
- проведением испытаний, проверкой инженерных обоснований (в том числе с привлечением независимых экспертов)
- Принятием осознанного решения "Go — No Go — Cancel" (пройти на следующую стадию жизненного цикла — вернуться и доработать — прекратить проект в целом). Если риск продолжения проекта приемлемый, то идём на следующую стадию. Если риск высок, то возвращаемся на доработку, снижающую риск. Если риск очень высок и доработка вряд ли поможет его снизить, то проект закрывается, жизненный цикл системы прекращается.

Desicion gate — это развилка в проекте, имеющая дело прежде всего с оценкой рисков его продолжения, и это нельзя забывать.




Обычно решение "Go" означает принятие решения по дальнейшему финансированию проекта (даже если деньги гарантированно были выделены на прохождение нескольких стадий жизненного цикла, в гейтах это выделение ресурсов пересматривается — проходит commitment review). Одна из методик управления жизненным циклом так и называется — ICM, incremental commitment model (пошаговое выделение ресурсов). Подробней см. http://ailev.livejournal.com/691464.html. Вехи — это просто дополнительные точки контроля, на которых проверяется выполнение графика работ, но не ожидается принятие решений "Go-NoGo-Cancel".

Важно понимать, что гейты и вехи — это в какой-то мере тоже стадии жизненного цикла, только не имеющие продолжительности (такая трактовка даётся в ISO 24744). Хотя это и не совсем так: в жизни гейты могут занимать ощутимое время и их тогда лучше считать отдельными стадиями жизненного цикла (чаще всего это происходит с гейтом "приёмка в эксплуатацию" — для крупных строек это может занимать до полугода, и если это обозначить "точкой на графике", то где предусмотреть ресурсы на проведение всех необходимых работ в эти полгода?).

Гейты и вехи позволяют договариваться менеджерам и системным инженерам: для

менеджеров они означают обычно даты принятия и исполнения бюджетных, ресурсных и других логистических решений, а у инженеров это означает сроки проведения инженерных мероприятий, выполнения всех необходимых для создания целевой системы работ.

### Рабочие продукты для определения жизненного цикла

Как необходимо определять (проектировать, конструировать) целевую систему, так необходимо определять (проектировать) деятельность обеспечивающих систем, т.е. жизненный цикл. Альфа определения (definition) жизненного цикла выражается в рабочих продуктах — описаниях (description) жизненного цикла, чаще всего это разного сорта диаграммы (простейшими из которых являются одномерные "стрелочки времени с зарубками на границах стадий" и "колбаски с именами стадий", более сложные представляются двумерными диаграммами, а самые сложные подразумевают использование графических языков ситуационной инженерии методов).

Вот пример диаграммы жизненного цикла, используемого консорциумом FIATECH в качестве roadmap (долгосрочного плана работ) по развитию отрасли капиталоёмких проектов (capital projects — строительство заводов, инфраструктурных сооружений типа мостов и эстакад и т.д.), эта диаграмма жизненного цикла разработана в 2004г:


Обратите внимание на то, что наименования части стадий даются по именам практик (отглагольные существительные), а часть по состояниям целевой системы на этой стадии. Кроме этого авторы диаграммы вынуждены были показать практики (управление проектом, управление данными) и ресурсы (новые материалы, рабочая сила и т.д.), которые должны быть использованы на всём протяжении жизненного цикла. Различные передачи рабочих продуктов (hand over) показаны стрелками, но

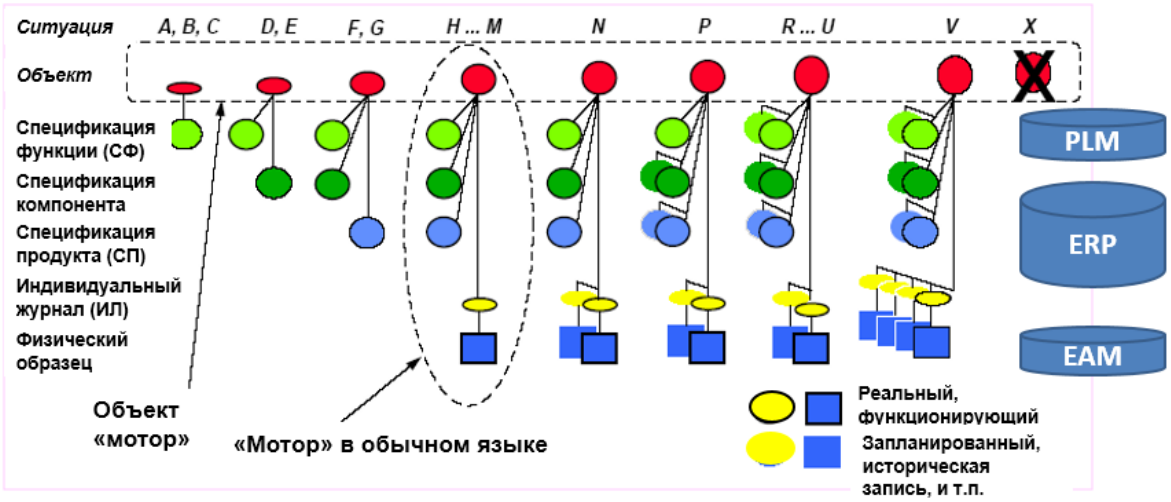
и природа этих стрелок различна (так, “обратная связь от опыта и знаний” относится вообще к разным системам из разных проектов: когда объект уже построен, сценарное планирование будет вестись уже для другого проекта). Очевидно, что для составления этой диаграммы не было использовано никакого метода описания жизненного цикла, но сама идея для рассказа о какой-то сложной деятельности иметь в основе жизненный цикл целевой системы этой деятельности — это правильная и продуктивная идея.

### Информационные системы управления жизненным циклом

Информационные системы управления жизненным циклом (PLM, product life cycle management) по факту поддерживают не полный жизненный цикл, а главным образом стадию проектирования. Хотя часто пытаются говорить о PLM как product life cycle management (практике управления жизненным циклом изделия/продукта/установки/сложного инженерного объекта/системы), аббревиатура PLM чаще всего используется для указания на вид инженерных информационных систем, выполняющих следующие функции:

- Управление конфигурацией инженерной системы на стадии архитектурного проектирования (хранилище информации проекта — PDM, product data management). Самые различные (механические, электрические, технологические и т.д.) САПР и системы инженерных расчётов работают со связанными между собой моделями в этом хранилище. Хранилище поддерживает версионирование моделей.
- Управление изменениями (отслеживание дел, главным образом запросов на изменение проекта/design), наиболее часто в виде issue tracker
- Формирование и передача информации конфигурации на следующие стадии жизненного цикла (прежде всего — выпуск спецификаций для закупки). Для этого в состав PLM входит генератор отчётов, берущий информацию из PDM.

Вот информационные системы в жизненном цикле мотора, появляющегося в инженерном проекте:




Сначала мотор появляется в виде компонента на принципиальной схеме (спецификация функции), затем у мотора появляются его характеристики, получающиеся в результате инженерных расчётов. До этих пор мотор — это “комплектующее” и информация о нём находится в PLM. Спецификация продукта — это спецификация модуля, который можно заказать по промышленному каталогу, о

нём известно только имя модели (но не серийный номер!). Модуль на стадии закупки — это уже "предмет снабжения", информация о нём, скорее всего, содержится в ERP-системе. Закупленный мотор-модуль монтируется, и становится "установленным оборудованием". Информация о нём (индивидуальный журнал для конкретного мотора с серийным номером, куда попадают записи о поломках, техобслуживании и т.д.) теперь находится в EAM системе (Enterprise asset management), используемой службой эксплуатации.

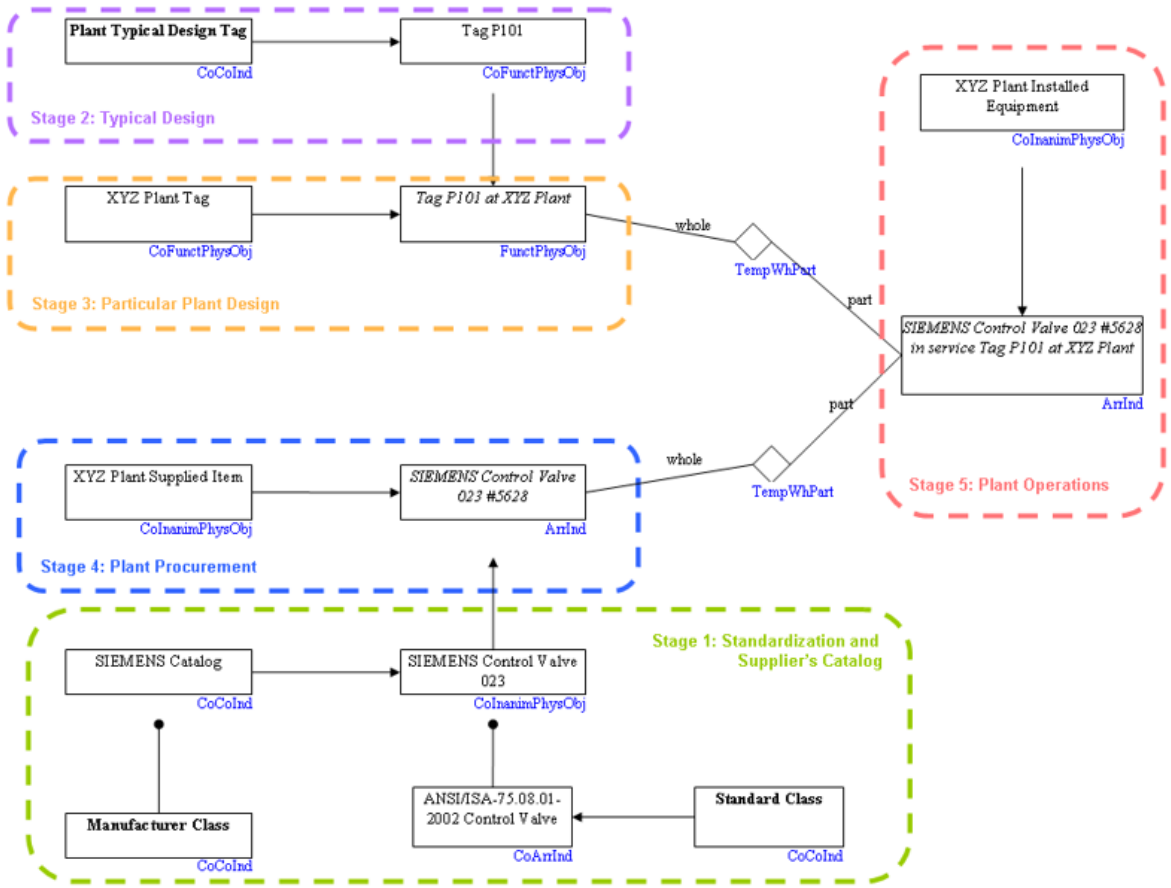
### Управление информацией/данными жизненного цикла

Информацией (данными) жизненного цикла называются данные, которые появляются, хранятся, передаются, обрабатываются и используются в любой момент жизненного цикла. Одной из важнейших практик управления информацией (данными) жизненного цикла является передача нужной информации (данных) между информационными системами различных организаций, обеспечивающих работу по ведущим практикам различных стадий жизненного цикла — hand over (эта передача подразумевает, что "меняется владелец", т.е. ответственность за управление конфигурацией переданной информации переходит тоже).

На практике это означает, что информация по мотору из предыдущего раздела будет перевведена руками в среднем 7 раз за время проекта. Это медленно, при этом вносятся ошибки, и это очень дорого (ибо работают не компьютеры, а люди). Ситуация осложняется и тем, что PLM, ERP, EAM системы часто находятся в разных организациях, ответственных за разные стадии жизненного цикла целевой системы (например, атомной станции или судна-контейнеровоза, куда должен быть установлен мотор из примера).

Основная задача управления информацией/управления данными — это то, чтобы информация/данные были доступны там и тогда, где и когда они нужны в ходе жизненного цикла. По факту это означает интеграцию информационных систем и их данных (сейчас всё чаще говорят "федерирование", подчёркивая факт независимости отдельных информационных систем и их данных). Это весьма нетривиальная задача, которая для своего решения требует привлечения особого рода специалистов: модельеров данных (они отличаются и от инженеров, и от программистов. Их задача — разработка структур данных, отражающих предметную область и реализующихся затем в конкретных базах данных. То есть они работают в тесном контакте с инженерами и программистами). Можно назвать модельеров данных "прикладными онтологами", ибо корректное и достаточно формальное, чтобы его можно было "объяснить компьютеру" описание мира является их основной задачей.

Вот пример модели данных (онтологии), отображающей жизненный цикл трубопроводной задвижки (Control Valve), описываемый примерно так, как описывался жизненный цикл мотора из предыдущего раздела. Это описание использует метод описания из ISO 15926:




Обратите внимание, что часть отношений между сущностями этой модели данных проходит через границы информационных систем, работающих на разных стадиях жизненного цикла установки (plant) — и значительная часть таких отношений будут отношениями Temporal Whole Part. (четырехмерный объект является временнОй частью четырехмерного объекта целиком, например, установленная, т.е.in service задвижка с серийным номером #5628 является полной темпоральной (временнОй) частью компоненты-задвижки P101 на установке XYZ и временнОй частью модуля Siemens Control Valve 023). Именно потому, что приходится работать со временем в условиях, когда описание одного и того же объекта в ходе жизненного цикла меняется разительно, стандарт ISO 15926 использует онтологию 4D экстенсионализма — система представляется не только в её уровнях декомпозиции, не только как совмещение разных объектов, которыми оперируют разные стейкхолдеры, но и как совокупность всех её состояний в ходе жизненного цикла. Моделирование инженерных данных должно это отражать.

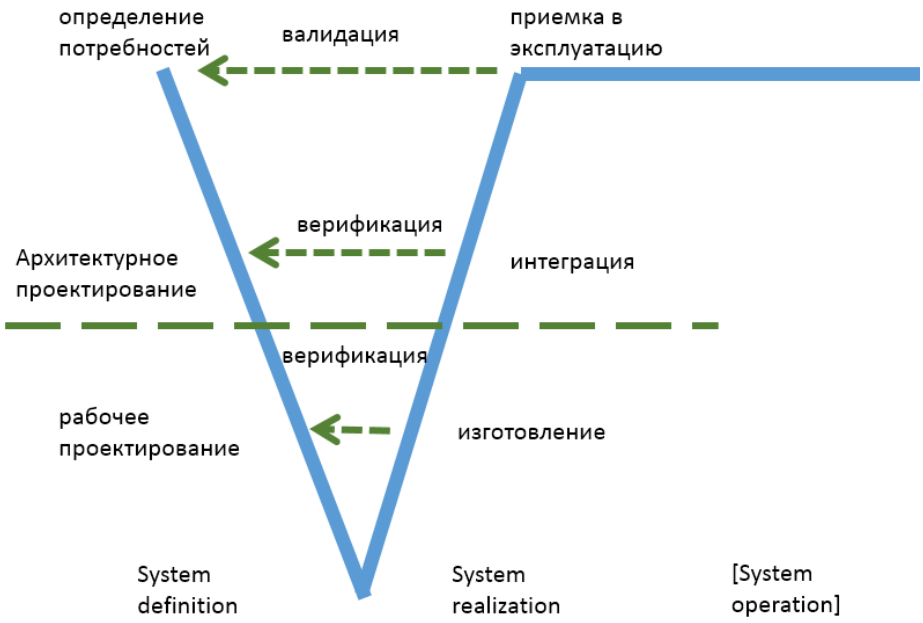
### *Практики жизненного цикла*

Практики (дисциплины + рабочие продукты + инструменты по их поддержке) часто называют практиками жизненного цикла, ибо выполнение этих практик и составляет основное содержание деятельности в инженерном проекте: инженерия требований, инженерия системной архитектуры, управление проектами и т.д. Обычно из числа практик жизненного цикла исключают практики, направленные на поддержание всего предприятия как стабильно существующего юридического лица — бухгалтерский учёт, например.

Учёт распределения практик жизненного цикла во времени возможен, если от “колбаски” жизненного цикла перейти к двумерным группам описания.

V-диаграмма

Самая знаменитая диаграмма жизненного цикла (да и всей системной инженерии) — это V-диаграмма. Иногда её называют V-model, поскольку рассматривают и как вариант вида жизненного цикла "водопадного" жёсткого стиля — просто "каскад" перегнут в точке "изготовления" (перехода от стадий определения системы к стадиям реализации системы).




Именно V-диаграмма используется чаще всего, чтобы пояснять самые общие черты системноинженерного процесса/метода/жизненного цикла:

- Фундаментальную разницу между практиками определения системы (работы с информацией), реализации системы (работы с веществами и полями), а также использованием системы. В том числе на V-диаграмме показывается основная идея системной инженерии "восемь раз отмерь, один раз отрежь": рекомендуется максимизировать трату ресурсов на более ранних стадиях, чтобы потом экономить трату много больших ресурсов на более поздних стадиях — с битами работать дешевле, чем с веществом, и поэтому ошибки легче исправлять в данных, чем в веществе!
- Соответствие определений и воплощений системы, поддерживаемое через проверки (верификация) и приёмки (валидация)
- Ведущие практики жизненного цикла (хотя начальная задумка была просто перегнуть "ступеньки каскада" в точке изготовления, сохранив последовательность ступенек во времени, наличие «обратных стрелок» верификации и валидации позволяет перенести акцент в ступеньках-стадиях-практиках на используемые практики (дисциплины-рабочие продукты-инструменты), а не на стадии жизненного цикла (вся деятельность, ограниченная периодом от начального момента времени стадии до конечного момента времени стадии).
- Разницу между системноинженерными практиками (выше пунктирной линии), имеющими дело с системой в целом и "обычными" инженерными практиками, имеющие дело с частями системы.
- Взаимодействие между практиками: работа идёт отнюдь не по той практике-стадии, которой соответствует точка времени на диаграмме! Нет,

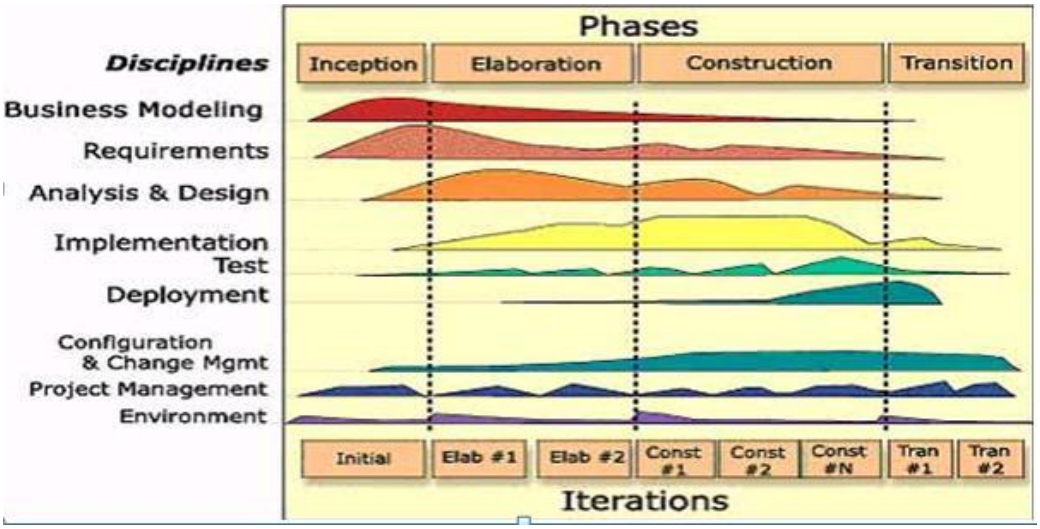
одновременно задействована вся "вертикаль" практик — архитектор общается и с инженерами по требованиям, и с занимающимися рабочим проектированием, а инженер-интегратор общается и с эксплуатационщиками, и с производителями оборудования.

Эта простейшая диаграмма имеет огромное число вариаций и модификаций (например, см. http://en.wikipedia.org/wiki/Dual_Vee_Model).

### Горбатая диаграмма

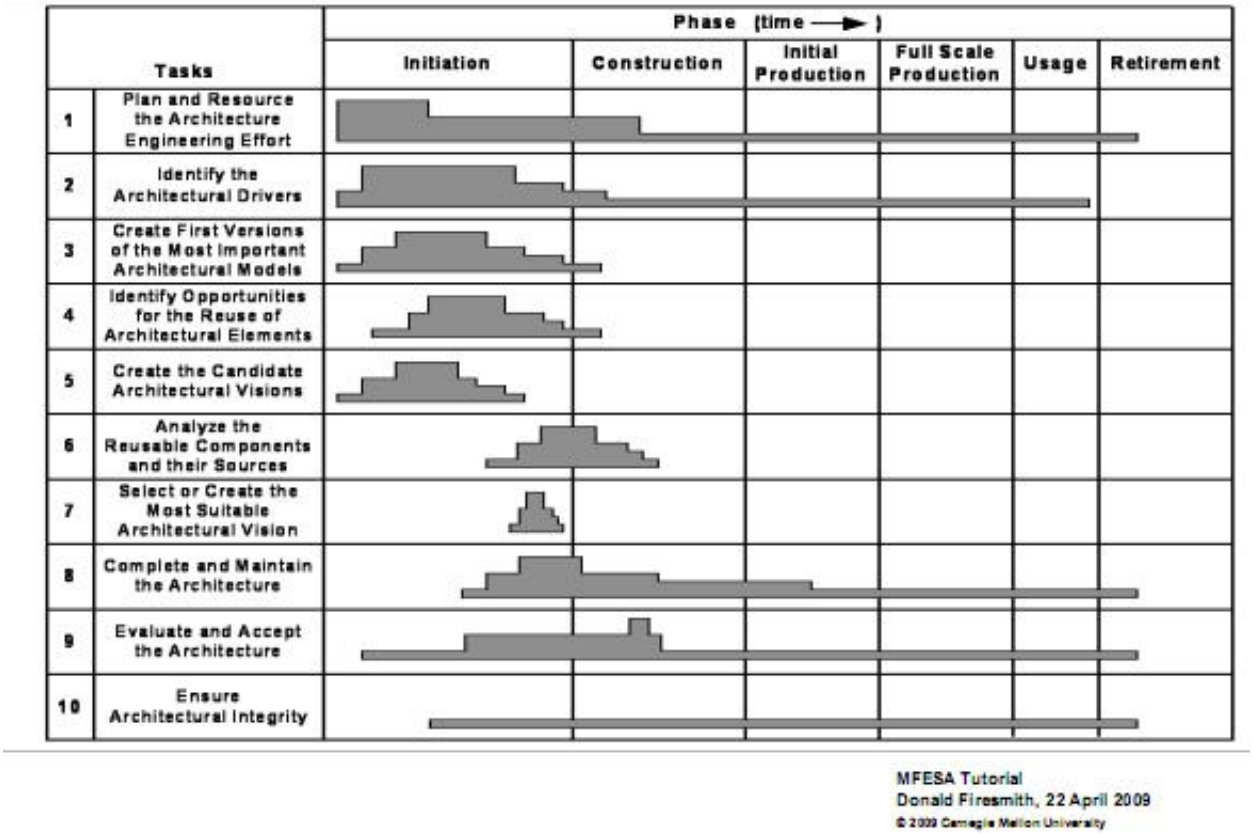
Горбатая диаграмма (hump diagram) обычно используется для того, чтобы показать относительные трудозатраты по различным практикам в ходе жизненного цикла. На этой (уже полностью двумерной, в отличие от "одномерной изогнутой" V-диаграммы) диаграмме уже чётко различаются практики (именуемые по их ведущей дисциплине) и стадии жизненного цикла (привязанные ко времени). Собственно, сама диаграмма придумана была для того, чтобы показывать количество времени, которое тратится на те или иные практики на тех или иных стадиях жизненного цикла — и эта диаграмма чётко показывает, что большинство практик оказываются выполняемыми на разных стадиях жизненного цикла, а одна стадия жизненного цикла включает выполнение многих практик.

Горбатая диаграмма для методологии RUP (Rational Unified Process):




На этой диаграмме phases (фазы) это стадии жизненного цикла, которые в свою очередь разбиты на итерации. Практики названы "дисциплинами" (и помним, что практики обычно называют именно по названиям дисциплин, а не по названию рабочих продуктов или инструментов, так что ничего тут странного — особенно, если учесть, что это диаграмма самого общего вида, а рабочие продукты и инструменты будут конкретизированы только в рамках конкретной организации, и даже конкретного проекта). Чётко видно, что работы по требованиям продолжаются вплоть до стадии передачи в эксплуатацию (transition), а тестирование начинается на начальных стадиях, а не только при подготовке к передаче в эксплуатацию.

Вот ещё один пример горбатой диаграммы: распределение труда по стадиям жизненного цикла в практиках архитектурной работы по MFESA:




Этот пример показывает, что hump диаграму можно рисовать не только для практик в целом (например, архитектурная работа в RUP), но и для подпрактик (пример разбиения архитектурной работы на её подпрактики/tasks в MFESA).

### *Водопад и agile*

Вид жизненного цикла

Жизненные циклы в плане распределения работ по разным практикам по времени бывают устроены очень по-разному. Об этом также и говорят по-разному:

- Виды жизненного цикла — мы будем использовать этот термин, если хочется непременно использовать термин "жизненный цикл", чтобы подчеркнуть наличие стадий.
- Формы жизненного цикла
- Метод разработки (development method). Напомним: метод — это прежде всего набор необходимых для работы практик, метод может не закрывать полного жизненного цикла). Полный синоним — методология разработки (development methodology). Слово "разработка" обычно означает стадии жизненного цикла вплоть до эксплуатации.
- Процессы разработки (development process), в том числе процесс разработки программных средств (software process).
- Модель жизненного цикла (life cycle model) — но не путайте с рабочим продуктом-моделью (моделью в значении "упрощённое представление системы")! Речь тут идёт просто об именовании варианта альфы определения жизненного цикла (т.е. использование слова "модель" как для указания моделей автомобилей: модель ВАЗ 2110, Ford Focus ST, или модели

фотоаппаратов: Nikon D3300, Sony SLT-A77 II, Olympus OM-D E-M10).

По большому счёту, это всё одно и то же: определение того, как будет устроена разработка (т.е. до стадии изготовление, иногда включая стадию изготовление) — какие практики, какие стадии (в том числе гейты), какие правила лежат в части выбора практик и распределения работы по стадиям.

Виды жизненного цикла, которые "на слуху" — это RUP (Rational Unified Processel), SCRUM, спиральная модель, и так далее: их сотни.

### Стили разработки: водопад и agile

Условно в видах жизненного цикла (методах разработки) можно выделить два больших стиля:

- Водопад (cascade, но переводят "каскад" редко, хотя такой перевод и более правилен)
- Agile (на русский обычно не переводится, хотя иногда говорят о "гибких методах")

К водопадным стилям относят такие, при которых отдельные практики выполняются последовательно, как вода проходит каскад — сначала разрабатываются требования, потом архитектура, потом проект, потом проходит изготовление, потом сборка и т.д. Все рабочие продукты стадий проходят hand over (передачу) на следующую стадию, и дальше не меняются, а используются для получения продуктов этой стадии.

Ключевая предпосылка "водопада" — это то, что практики совпадают со стадиями жизненного цикла. То есть "проектирование" — это и практика, и стадия жизненного цикла. "Тестирование" — это и практика, и название жизненного цикла.

Рисуют это примерно так (оригинальная статья Royce, Winston (1970), «Managing the Development of Large Software Systems» — http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf):

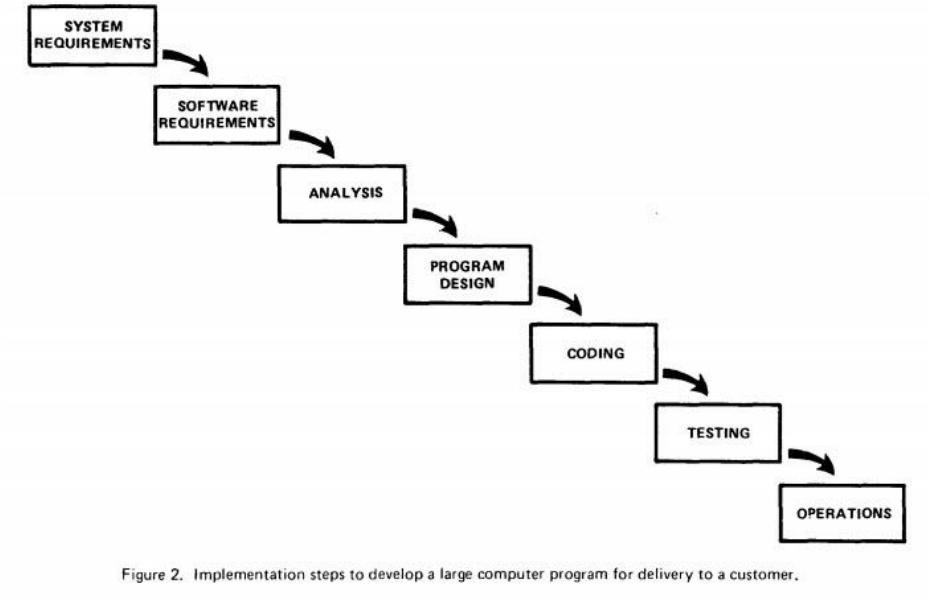
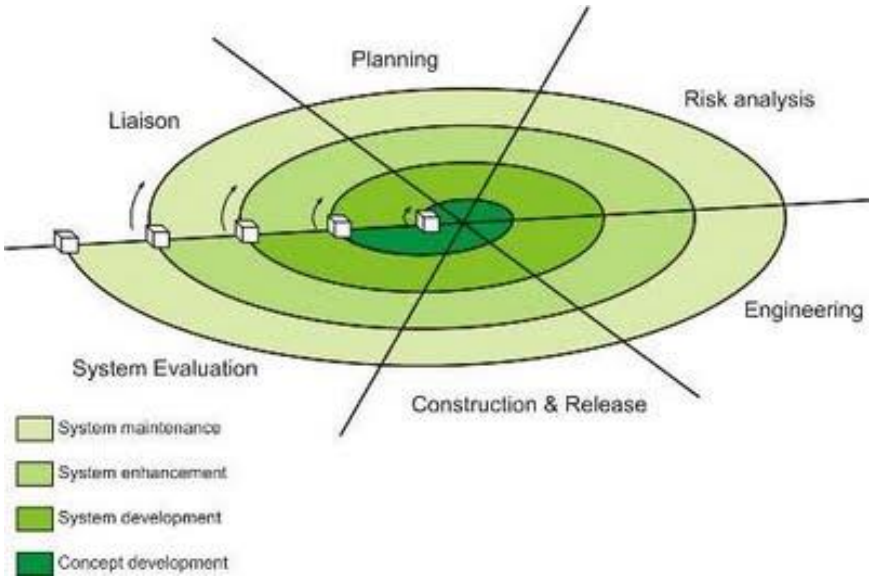



Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

К сожалению, эта картинка показывает фантастическую ситуацию. В реальной жизни выполнение работ каждой практики-стадии вызывает необходимость обращения к уже прошедшим практикам (но их стадии-то прошли, и ресурсов на

них уже нет, и рабочие продукты уже считаются сданными!). Ужас в том, что возвращаться назад приходится часто не на одну стадию-практику-ступеньку, а сразу на несколько, в том числе бывает, что и с последней на первую!

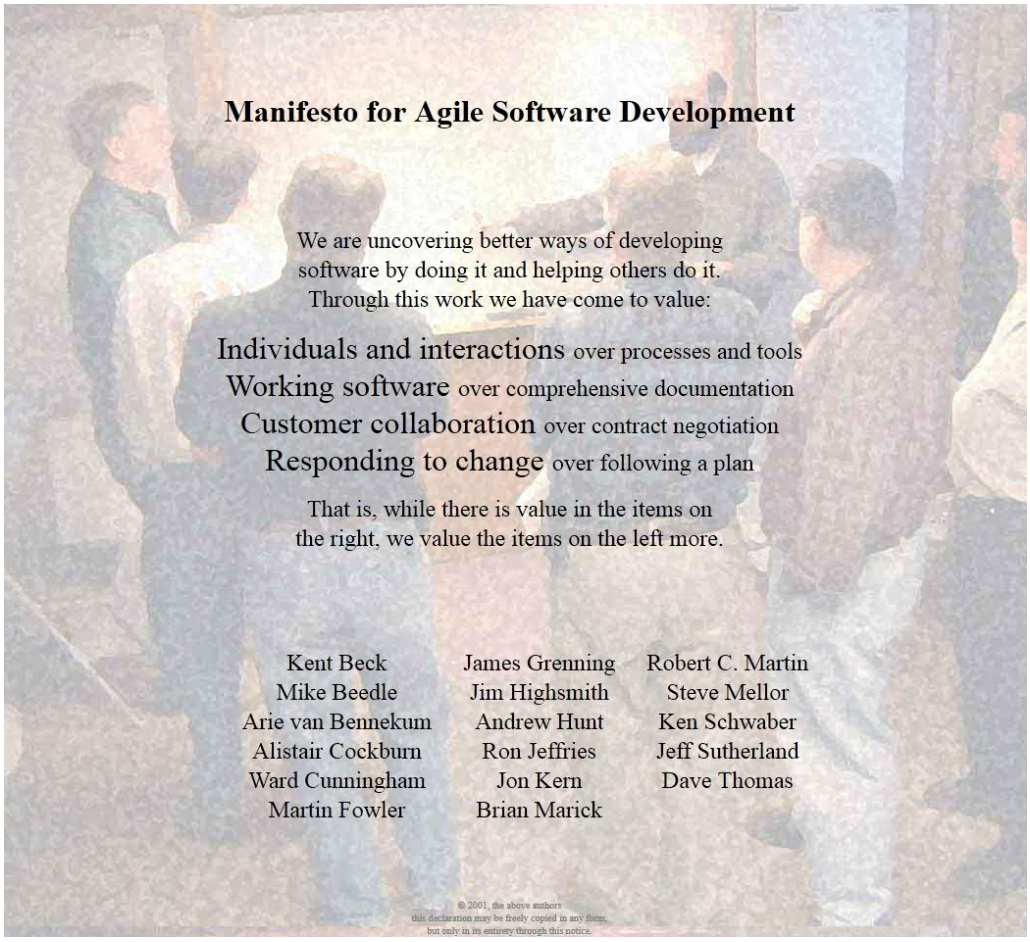
Для борьбы с этим были придуманы альтернативные стили, самой популярным из которых был придуманный Barry Boehm стиль "спиральной" модели жизненного цикла — в которой говорилось, что разработка происходит "по спирали", в которой одновременно выполняется цикл из нескольких практик, поэтому практики и стадии жизненного цикла оказались разъединены. Вот типичное изображение спиральнуой модели (http://mydotnetcoolfaqs.blogspot.ru/2011/04/software-development-life-cycle-sdlc.html — и по этой ссылке ещё некоторое количество древних моделей жизненного цикла):




Разными оттенками зелёного показаны стадии жизненного цикла (разработка концепции, разработка системы, усиление системы, сопровождение/обслуживание системы), а практики показаны как сектора на круге (liason тут — customer communications).

Проблема оставалась: практики предполагались последовательно выполняющиеся на каждом витке спирали, т.е. сохранялся "микроводопад".

Потом было предложено огромное количество других моделей/видов жизненного цикла, пытающихся снять недостатки "водопада", но в 2001 году группа программистов (software engineers) предложила Agile manifesto (http://agilemanifesto.org/):




Официальный русский перевод (http://agilemanifesto.org/iso/ru/):

## Agile-манифест разработки программного обеспечения

Мы постоянно открываем для себя более совершенные методы разработки программного обеспечения, занимаясь разработкой непосредственно и помогая в этом другим. Благодаря проделанной работе мы смогли осознать, что:

Люди и взаимодействие важнее процессов и инструментов
Работающий продукт важнее исчерпывающей документации
Сотрудничество с заказчиком важнее согласования условий контракта
Готовность к изменениям важнее следования первоначальному плану

То есть, не отрицая важности того, что справа, мы всё-таки больше ценим то, что слева.

После этого манифеста началась “религиозная война”: по сути, манифест полностью разделял стадии жизненного цикла и практики, а также предполагал “гибкость” в планировании работ — то есть объявлял отсутствие “водопада” в любых его проявлениях.

Появились сотни вариантов методов разработки, из которых сегодня для программной инженерии наиболее распространён SCRUM.

На практике оказалось, что следование принципам agile методов разработки и предлагаемым ими моделям жизненного цикла (чаще всего делящих жизненный цикл на стадии "релизов" с выделением мелких подстадий "итераций") может приводить как к впечатляющим успехам, так и к впечатляющим провалам. Также оказалось, что и "водопад" в чистом виде никто не использует, поскольку "по документам" работа описывалась одним образом, а "в жизни" элементы agile использовались много больше. Много было и обычного непонимания. Так, "люди и взаимодействие важнее процессов и инструментов" — это вовсе не был призыв к луддитству и отказу от инструментов. Просто не было ещё инструментов, которые поддерживали agile-методы разработки. В итоге перестали использоваться как чисто гибкие методы (на их использование ещё и менеджеров было трудно уговаривать, потому как они не дают хорошо спланировать работу заранее и потом контролировать выполнение планов), так и "водопады" (ибо жёстко запланированные проекты часто получаются дороже и длятся дольше, чем было запланировано — планы обычно не включают неожиданностей типа уточнения требований близко к концу разработки или переделки архитектуры после проведения испытаний уже готового изделия).

В системной инженерии "железных" систем методы agile пока используются мало, ибо принято считать, что в существенной мере действует "водопадная" модель. Но в последние годы ситуация быстро меняется: разработка определения системы в существенной мере оказывается похожей на разработку программного обеспечения, и в ней для разработки моделей могут быть использованы те же практики, которые хорошо зарекомендовали себя при разработке программного обеспечения. Единственное что, так это временной лаг: модные в программной инженерии идеи попадают в системную инженерию с задержкой в примерно 10-15 лет. Как раз сейчас начинают говорить о том, что какие-то agile элементы в системной инженерии возможны. Но вот воплощение системы обычно проходит в стиле "водопада": если хорошо проработан проект, и имеется большой опыт создания аналогичных систем, то неожиданностей и "возвратов назад" в проекте будет мало, сборка проходит в точном соответствии с планом, испытания проходят с первого раза.

Хороший пример того, как agile проникает в системную инженерию — это организация хода разработок в компании SpaceX (см. https://www.aiaa.org/uploadedFiles/Events/Conferences/2012_Conferences/2012-Complex-Aerospace-Systems-Exchange-Event/Detailed_Program/CASE2012_2-4_Muratore_presentation.pdf). Главная там фраза — focus on tools not rules. А потом test rigorously and often.

Ещё один хороший пример — agile производство автомобиля: видеорассказ — http://www.youtube.com/watch?v=x8jdx-lf2Dw, описание автомобиля http://wikispeed.org/the-car/, eXtreme manufacturing — http://scrumlab.scruminc.com/articles.html/_/open/extreme-manufacturing-r93, test driven developing for hardware — http://wikispeed.org/2013/07/tdd-for-hardware/. Главное там было — обеспечить высокую скорость изменений, и указывались примерно те же механизмы, что и в SpaceX (обмен информацией через сеть вместо традиционных control boards, даже была оговорка, что ещё лет пять назад это было бы невозможно по причине недоразвитости сетевых сервисов, а лет десять назад всех этих сервисов вообще не существовало).

Близко к agile примыкает lean systems engineering: http://www.lean-systems-engineering.org/downloads-products/lean-enablers-for-systems-engineering/downloads-

lean-enabler-for-systems-engineering/. В принципе, это всё бесполезно читать, если вы не прочли про «непереводимое» на русский lean manufacturing (наиболее точно тут будет «не делать ничего лишнего»): http://en.wikipedia.org/wiki/Lean_manufacturing. Но и с прочтённым тоже может быть не просто.

Пример связи подходов lean и agile может быть найден в книге «Kanban and Scrum. Making the most of both» (это для софта, но там довольно популярно изложено несколько идей общего вида. Русский перевод!): http://www.infoq.com/resource/news/2010/01/kanban-scrum-minibook/en/resources/KanbanAndScrum-Russian.pdf

Когда все стадии жизненного цикла выполняются одновременно, то это называется concurrent engineering и по духу близко к agile – на минимальном уровне об этом рассказывается тут: http://incose-ru.livejournal.com/45719.html. Но по-настоящему впечатляющий пример – как стадион разбили на 12 секторов, и пока один сектор проектировали, предыдущий строили, а пред-предыдущий отделывали: http://www.myvi.ru/watch/WxsZCtYxMEyyp5QLEIp_Aw2 (6й сезон 14й фильм из замечательной серии телевизионных документальных фильмов про мастерство инженеров — http://en.wikipedia.org/wiki/Extreme_Engineering).

Главное тут сообразить, что «agile» и «без планирования, без особых принципов, как бог на душу положит» — это про разное. Agile подразумевает строгие принципы, строгую дисциплину, вполне определённые практики и использование специализированного инструментария для поддержки работы. И, как минимум, должно быть чёткое понимание, какие практики планирования и выполнения работ вы используете, как взаимосвязаны стадии работ, что вы делаете параллельно и что последовательно, в какой момент сотрудники или подрядчики синхронизируют выполнение своих работ.

### Паттерны жизненного цикла

Конечно, оппозиция "водопад vs agile" слишком груба. В зависимости от распределения различных рисков по стадиям жизненного цикла вместо обсуждения водопадных и agile стилей в работе http://csse.usc.edu/csse/TECHRPTS/2009/usc-csse-2009-502/usc-csse-2009-502.pdf предложили различать следующие паттерны:

- Купи готовое (Use Single NDI),
- Гибкий (Agile),
- Гибкий с архитектурой (Architected Agile),
- Формальные методы (Formal Methods),
- Оборудование с программными компонентами (Hardware with embedded Software component),
- Неделимость для начала эксплуатации (Indivisible Initial Operational Capability),
- Много закупок (NDI-intensive) — проектирование (в отличие от конструирования)
- Гибрид гибкости и плана (Hybrid agile/plan-driven system),
- Много собственников в системе систем (Multi-owner system of systems),
- Семейство систем (Family of systems),

- Brownfield (модернизация)
- Акцент на сервисах (Services-Intensive)

Нет никакой возможности обойтись одним и тем же вариантом жизненного цикла для разных типов проектов. Каждый проект должен получить свой индивидуальный метод разработки, индивидуальную модель жизненного цикла. Нужно также запомнить, что даже при повторении одного и того же проекта жизненные циклы будут разные — ибо первое выполнение проекта даст ответы на многие вопросы по рискам, и приобретённый опыт позволит изменить вид жизненного цикла для второго проекта в их серии.

### *Основной жизненный цикл*

#### Состояния альф

Каждая альфа инженерного проекта меняется во времени в его ходе: она проходит по графу состояний. Например, альфа "работа" проходит следующие состояния:

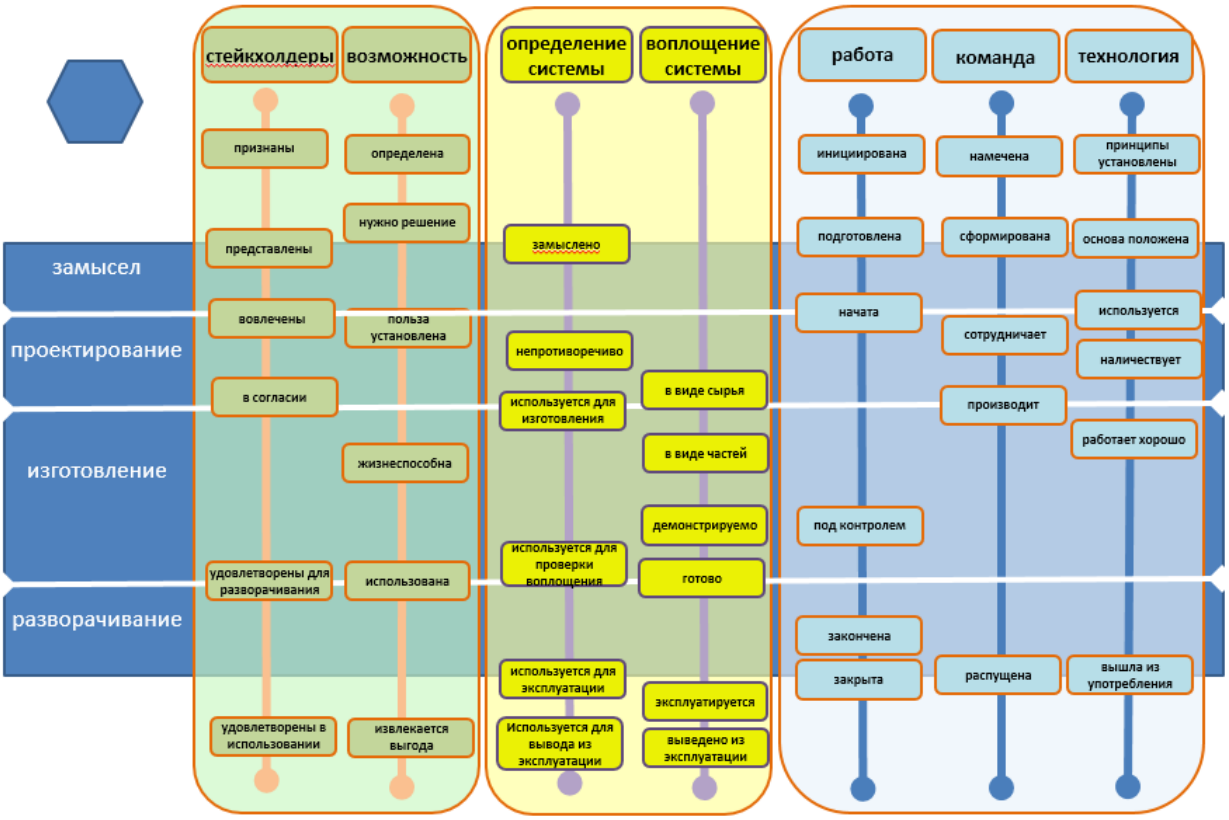
- Инициирована — работа была запрошена.
- Подготовлена — все предусловия для начала работы выполнены.
- Начата — работа происходит.
- Под контролем — работа продвигается хорошо, риски под контролем, уровень производительности достаточен для достижения удовлетворительного результата.
- Закончена — работа по производству результата была закончена.
- Закрыта — все остающиеся служебные задачи были завершены, и работа была официально закрыта. Полученные уроки были сформулированы, записаны и обсуждены.

Конечно, такая последовательность только гипотетична. В жизни альфы могут проходить через состояния циклически, какие-то состояния могут пропускаться, могут существовать альтернативные пути через другие состояния, даже не показанные на этом "счастливом пути" (happy path) по графу состояний. Граф состояний может иметь весьма сложную и запутанную структуру. Но мы будем существенно упрощать представление состояний, показывая только ситуацию прохода по "счастливому пути" — без возвратов, пропусков и альтернативных состояний.

Работы проекта как раз и проводятся для того, чтобы изменять состояния альф. Так, нужно попотеть, чтобы работа была в состоянии "под контролем".

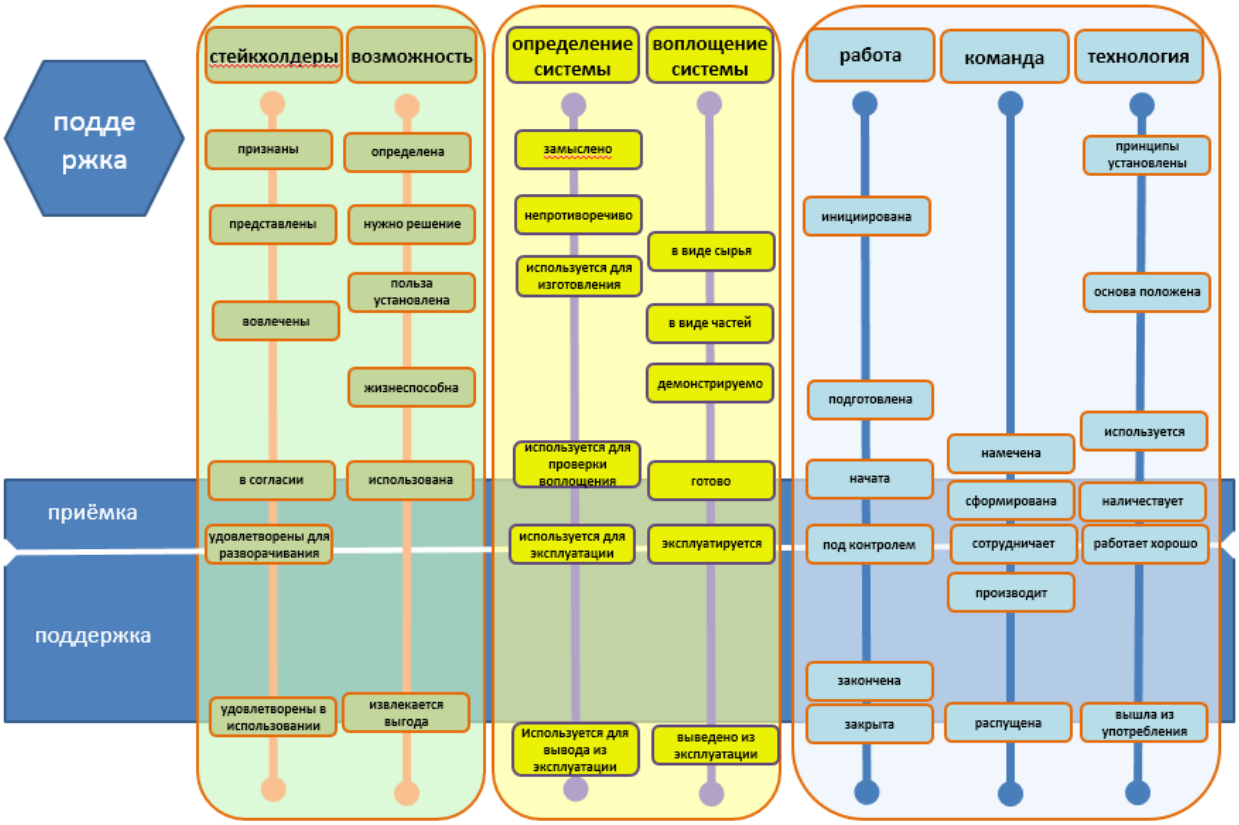
#### Основной жизненный цикл

Важно понимать, что в ходе инженерного проекта меняется отнюдь не только состояния альф инженерного решения (определения и воплощения системы) и их подальф (требований, архитектуры, компонент, модулей и т.д.). Нет, меняется состояние всех альф. Если взять основные (kernel из Essence — оба слова можно перевести как "основа") альфы инженерного проекта, то можно назвать его основным жизненным циклом. Вот как можно его представить для какого-то более-менее типового проекта:




Обратите внимание, что на диаграмме видны жизненные циклы и проекта, и системы — хотя тут трудно говорить собственно о целевой системе в части её определения и воплощения, ибо жизненные циклы показаны также и для остальных альф. Хорошо видно, что часть жизненного цикла системы проходит до начала проекта, а часть будет происходить после.

Если взять какой-нибудь проект по сопровождению системы, то эта разница между жизненным циклом системы и жизненным циклом проекта будет ещё более видимой:




Задача управления жизненным циклом тем самым заключается не только в том, чтобы альфы воплощения и определения системы продвигались по своим состояниям, но и в синхронизации достижения состояний всех основных альф инженерного проекта. Синхронизация состояний альф кладётся в основу планирования работ по проекту, в план работ включаются и работы по не-инженерным альфам (о чём часто забывается при планировании, и вследствие этого при обеспечении плана ресурсами: в инженерных проектах очень часто существенно не хватает менеджерских работ, менеджеры и операционные, и клиентские перегружены — и от этого страдает весь проект в целом).

Определение жизненного цикла является главной альфой (т.е. отслеживаемым в ходе инженерного проекта объектом), по которой происходят договорённости между менеджерами и инженерами. Это подальфа “определения системы” (ибо определение жизненного цикла связано в существенной мере с работой), а также “работы” (ибо лежит в основе работ по планированию проекта) и “технологии” ибо вид жизненного цикла определяет используемые практики жизненного цикла.

### Практики жизненного цикла в версии ISO 15288

Управление жизненным циклом охватывает и инженерные практики, и практики маркетинга/стратегирования, и практики менеджерские. Отслеживать продвижение только определения и воплощения системы, а также практиковать только связанные с ними практики опасно для инженерного проекта, и это получает отражение в стандартах системной инженерии. Вот набор практик жизненного цикла системной инженерии из ISO 15288:

1. Контрактация
1.1. Закупка
1.2. Поставка

2. Обеспечения проектов

2.1. Определение вида (описывание, моделирование) жизненного цикла
2.2. Управление инфраструктурой
2.3. Управление портфелем проектов
2.4. Управление персоналом
2.5. Управление качеством

3. Проектные
3.1. Управление проектами
3.1.1. Планирование проекта
3.1.2. Управление выполнением и контроль проекта
3.2. Поддержка проектов
3.2.1. Управление решениями
3.2.2. Управление рисками
3.2.3. Управление конфигурацией
3.2.4. Управление информацией
3.2.5. Измерения

4. Технические
4.1. Сбор требований
4.2. Анализ требований
4.3. Архитектурное проектирование (дизайн)
4.4. Изготовление
4.5. Верификация (проверка)
4.6. Переход к эксплуатации
4.7. Валидация (приёмка)
4.8. Эксплуатация
4.9. Обслуживание
4.10. Вывод из эксплуатации

С целевой системой в плане продвижения альф определения и воплощения системы непосредственно работают главным образом технические практики из ISO 15288. Остальные практики жизненного цикла системной инженерии работают с обеспечивающей системой, продвигая альфы работы, технологии, команды, возможностей и стейкхолдеров. Само определение вида жизненного цикла входит как отдельная практика (2.1).

## 9. Практика контрольных вопросов

*Контрольные вопросы для управления жизненным циклом*

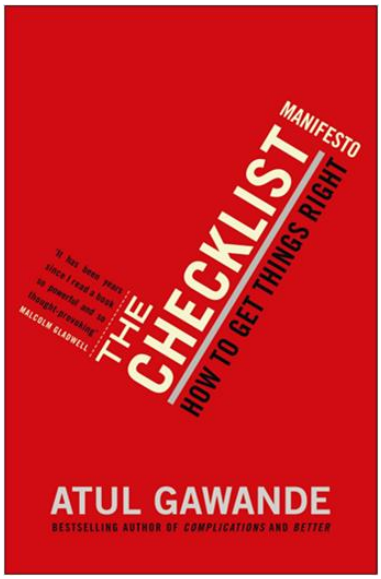
Успех контрольных вопросов

Человеческая деятельность становится сверхсложной до такой степени, что одним виртуозом уже не выполнима, как бы ни был велик этот виртуоз в своём мастерстве. В силу сверхсложности этой деятельности на каждого виртуоза бывает проруха: пропуск какого-нибудь из самых существенных (базовых, известных наизусть всем, невозможных к пропуску, критических для успеха) шагов в выполняемых ими работах. Просто по общей затурканности сверхспециалистов, у которых в голове ой-ой-ой сколько всякого, нужного для выполнения работы.

Сверхспециалистов в проекте много разных, и критическими шагами является не просто выполнение процедур, но и простое знакомство друг с другом (ибо в каждом

даже простом действии встречаются ранее незнакомые друг с другом люди, не имеющие опыта совместной работы), а также знакомство друг друга с выполнением этих базовых и критических для успеха всего проекта действий, выполненных каждым человеком. Часто происходит так, что все члены команды уверены, что кто-то выполнил важнейшее, но тривиальное и банальное действие — но оказывается, что его никто так и не выполнял, некогда было всем!

Одна из самых мощных практик, позволяющая снизить фатальные ошибки от досадных промахов в сложной деятельности — это практика контрольных вопросов. Она лучше всего описана в книге Atul Gawande "The Checklist Manifesto", http://gawande.com/the-checklist-manifesto (перевод на русский: http://www.alpinabook.ru/catalogue/2118996.html). Её суть: "мало что более эффективно в сверхсложных коллективных проектах, чем дисциплинированно проходимые чеклисты, но это мало где признаётся, кроме авиации, космонавтики, медицины".




Контрольные вопросы (checklist, “список контрольных вопросов”, а “отдельный вопрос” будет “checkpoint”) существуют в самых разных формах: от “классического” авиационного, космического или хирургического списка контрольных вопросов до списка работ в строительном проекте в составе структуры разбиения работ, от списка проверок due diligence в инвестиционных фондах до таблички на дверях “выключил ли ты свет?”. Чеклист обычно — это короткий список (пять-семь, и лишь иногда семьдесят или семьсот) действий, каждое из которых должно быть сделано во время зачитывания списка (чеклист типа READ-DO), или результатов/условий/действий, которые должны быть подтверждены по мере зачитывания списка (DO-CONFIRM).

Использование чеклиста происходит во время "точек паузы" (pause points, как они называются в авиации), где вся рутинная и сложная деятельность прекращается, и происходит прогон чеклиста. При ближайшем рассмотрении оказывается, что наиболее эффективно использование чеклиста при смене состояний каких-то альф, в которых принимается решение Go-NoGo в ходе жизненного цикла альфы, т.е. в точках возможного “невозврата”. Чеклист "прочти-подтверди" (DO-CONFIRM) представляет собой проверку того, что сделано на предыдущей стадии, чтобы попасть в следующую стадию. Ну, или чтобы попасть в следующую стадию, нужно прогнать чеклист "прочти-сделай" (READ-DO, этот тип чеклиста обычно выполняется при попадании во внештатную ситуацию). Поэтому в каждом деле

главных чеклистов обычно буквально несколько штук, а вот "аварийных" могут быть сотни.

Прохождение одного "классического" подтверждающего чеклиста (опять же — недаром его выполнение происходит в точке "паузы"!) занимает одну-две минуты нужно вслух подтвердить для всего коллектива выполнение указанных в нём условий (типа "снято с ручного тормоза", "выпуск релиза происходит не вечером пятницы", "антибиотик введён не раньше часа до операции, и не позже момента начала операции"), а самих условий обычно 5-9. Конечно, "неклассические" варианты отличаются разнообразием, особенно в случаях, когда речь идёт об использовании графика проекта в качестве чеклиста, чтобы иметь уверенность в том, что работы по каждому пункту были действительно выполнены и ничего не забыто.

Дисциплина чеклистов выглядит очень бюрократической — но это ровно то же самое, что отличает системных инженеров, которых совершенно недаром с их приверженностью к формальным процедурам (aka прогону чеклистов) называют "бюрократами от инженерии", что они признают, и чего они не стесняются. Объясняют это своё нестеснение они просто: зато ракеты долетают куда надо, а атомные станции начинают давать ток и даже более безопасны в целом, чем угольные в расчёте на киловатт-час.

Чеклисты, тем не менее, существенно отличаются от похожих на них описаний работ (например, описанных по традициям ситуационной инженерии методов, или традициям архитектуры предпринятий, или традициям описания лучших практик, или просто пошаговых инструкций по выполнению работ). В чеклистах приводится не 100% описания работ и не цитируется содержание учебников и регламентов. В чеклистах спрашивается только самое существенное (и обычно “банальное”), что может быть упущено и приведёт к epic fail. Так, первый пункт чеклиста, выполняемого при остановке двигателя самолёта — “1. Fly the aircraft”. Все знают, что нужно не забывать управлять самолётом, но когда у самолёта останавливается двигатель, лётчик начинает интересоваться двигателем, топливом, состоянием электропроводки и забывает управлять самолётом. Потеря управления при остановке двигателя является причиной бОльшего числа аварий, чем собственно остановка двигателя! И это при том, что лётчики обычно хорошо тренированы! То же можно сказать и про врачей, проходящих серьёзный профессиональный тренинг. В литературе приводится цифра, что использование чеклистов, предлагаемых Всемирной организацией здравоохранения для хирургических вмешательств снижает риск смерти вплоть до 47%! И это без изменения самой техники хирургических вмешательств, без изменения используемых медикаментов или организации операции — только за счёт использования чеклистов, предотвращающих “глупые” ошибки (один из вопросов, например, “проверить, на какой стороне тела проводится операция”).

То же самое происходит и в проектах системной инженерии. Если есть проблемы с какой-то альфой проекта, то очень вероятно, что не выполнятся какие-то банальные процедуры (типа “забыли провести испытания” или “давно не общались со стейкхолдерами”). Лучше бы вовремя вспомнить о таких важных вещах, чтобы потом не иметь проблем из-за отвлечения внимания.

Главным образом чеклисты делаются на основе анализа статистики, а в более редкие "аварийные" чеклисты шаги включаются на базе расследований крупных катастроф. Ибо 10% провалов происходит не из-за чрезвычайной сложности какого-то процесса, а именно из-за того, что очевидные шаги пропускают на основе

того, что "всё тут обычно бывает гладко", а то и банально из-за отвлечений, надежды, что кто-то другой это сделал, общей суеты вблизи смены стадий жизненного цикла, плохого знакомства команды друг с другом. Теория швейцарского сыра (http://en.wikipedia.org/wiki/Swiss_cheese_model) — это как раз про чеклисты. Утверждается, что в 9 случаях из 10 чеклист будет пройден без сучка и задоринки, а в одном из десяти случаев будет что-то обнаружено, что исправится за несколько секунд, но убережёт от дорогостоящих и трудно исправляемых ошибок.

Чеклистам доверяют и их рутинно выполняют — как чистят зубы, как одевают уличную обувь, на уровне рефлекса. Их выполняют по сути, их не сачкуют, хотя и известно, что выполнять их бессмысленно в девяти случаях из десяти. Но в одном из десяти случаев ошибка таки выявляется, и этот честный прогон чеклиста окупает всё то время, что было на него потрачено во всех остальных прогонах: исправлять ошибки в разы дороже и затратнее, чем читать чеклисты.

Кровью написаны не уставы, а чеклисты. Ибо уставы/стандарты/нормы наизусть выучить — дело бесполезное, и требует сверхгероя. Никто не способен прочесть всю нормативную документацию, запомнить все требования, освоить пару десятков учебников ("уставы") так, чтобы обязательно их все вспомнить в критической ситуации. Но короткий чеклист имеет шанс быть вытащенным и прогнанным — это не заменяет учебников, требования высокой квалификации людей и непрестанного тренинга, стандартов, регламентов, инструкций и прочих обязательных к исполнению “уставов” (чтобы было кого наказывать за их несоблюдение, и чтобы было где-то хранилище полного знания). Но чеклисты позволяют из сверхгероев делать просто героев, а потом и рутинизировать процесс. По мере внедрения в инженерную и лётную практику чеклистов особость генконструкторов перестала волновать, и их именами перестали называть самолёты, пилоты-испытатели перестали быть национальными героями. А сейчас примерно то же самое происходит с медицинскими светилами (и об этом подробно рассказывает в своей книге Atul Gawande — эпоха Master Builders заканчивается, на смену ей идёт эпоха командной работы).

А уставы и чеклисты отличаются хотя бы тем, что уставы требуется выполнять полностью в их целостности, а чеклисты — прогонять (например, проговаривать вслух вопросы и проговаривать вслух ответы — причём в присутствии всей команды), прогонять всегда.

Важно прохождение чеклиста всей командой, это средство убедиться, что "все на одной странице книги", "играют одну и ту же партию", имеют "одну и ту же версию правды". При прохождении чеклиста важен как раз консенсус, что все члены команды (а часто и внешние стейкхолдеры) согласны с тем, что чеклист пройден, и принимается решение "go — no go" для возможного перехода к следующему состоянию альфы.

В самом чеклисте проверяется/делаются в обязательном порядке не только производственные акты (преобразуется информация и вещество), но и коммуникационные акты (поручения, обязательства, обещания, отчёты о выполнении и т.д. — то, что связано с выполнениями людьми своих полномочий). Чеклисты являются важным инструментом командообразования: вплоть до обязательно представления команды друг другу перед началом работы и требования вот прямо сейчас, во время точки паузы и прогона чеклиста предъявить план действий на следующей стадии, а также рассказать о подозрениях каждого члена команды о возможных трудностях в ходе реализации плана. Важно, что

зачитывание чеклиста производится не самым главным в команде, а самым свободным — но имеющим достаточный статус, чтобы мочь возразить главному при "проскоке" каких-то шагов в выполнении проверок при зачитывании чеклиста.

Короткий список критических шагов (или критических выполненных условий) должен быть извлечён из кортекса супергероев и перенесен в экзокортекс (т.е. записан на бумажку, или засунут в компьютер). Чеклисты — это одна из мер по повышению успешности решения сверхсложных задач не за счёт супертренинга супермозга супергроссмейстеров, а за счёт введения дисциплины коллективной деятельности и усиления кортекса в виде экзокортекса. Гроссмейстеры играют в уме, простые игроки играют за доской, "записывая" на ней результаты ходов — чтобы у обоих игроков не было ни малейших сомнений, что они играют одну и ту же партию, а не разные. Чеклисты позволяют синхронизировать командное понимание по поводу важнейших шагов деятельности, в разы и разы уменьшая количество ошибок.

Футбольное "порядок бьёт класс" вполне переносимо в инженерные проекты. Наличие супермастера не гарантирует от глупых ошибок, особенно в ситуации, когда таких супермастеров набирается суперкоманда — мало какая деятельность сейчас осуществляется в одиночку, а ошибки любят гнездиться "на стыках" действий разных людей. Чеклисты как раз про один из способов организации такого "порядка", который будет бить "класс". Чеклист не снижает свободу творчества, но ставит хоть какую-то защиту от глупых ошибок. Так сказать, дешёвая страховка для обычных смертных и их команд.

Что ещё похоже в "чеклистоведении" и системной инженерии — это отсутствие внимания к проблеме предотвращения глупых ошибок и интереса профессионалов к работе с чеклистами, несмотря на драматические улучшения (по сравнению с любыми другими менеджерскими и инженерными новинками) в поднятии качества работы. Мало кто верит в мощь чеклистов, а зря.

### Контрольные вопросы к состояниям альф

Альфы в ходе жизненного цикла меняют свои состояния, проходя жизненный цикл. Контрольные вопросы для состояний альф сформулированы по типу “READ-CONFIRM”, т.е. нужно ответить на утверждения “да”, чтобы подтвердить достижение какого-то состояния альфы. Эти утверждения кажутся банальными, но есть подвохи:

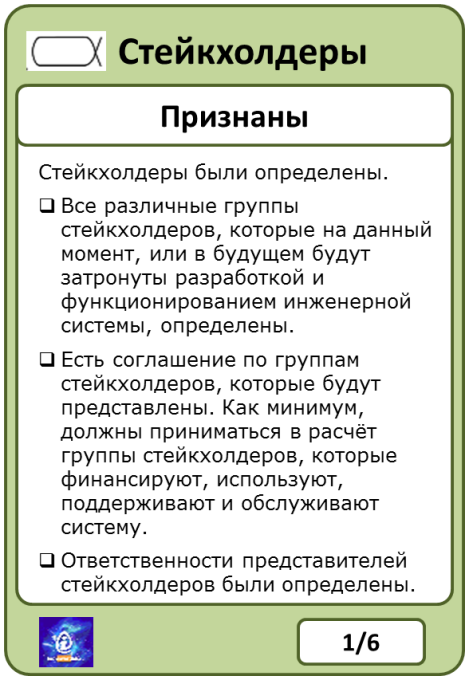
- Помним про пункт “1. Fly the aircraft!”. При всей примитивности и очевидности, утверждения должны быть проверены и ответы должны быть обоснованы. Если кто-то из членов команды знает, что ответ “нет”, то он *обязан* предупредить всех членов команды о реальном положении дел. Вся процедура коллективной читки чеклиста ровно на это и нацелена: если о проблеме случайно знает кто-то из членов команды, но не знают остальные — это знание будет доведено до всей команды, и можно будет предпринять какие-то меры. Если о проблеме умолчать, то меры не будут приняты, и риск провала проекта увеличится.
- На некоторые вопросы вы не сможете ответить, ибо не будете знать соответствующей практики, а иногда даже и просто дисциплины (так, вы не сможете ответить “да” на утверждение “механизмы управления конфигурацией описаний согласованы”, если вы не знаете, что такое “управление конфигурацией”. Вы не сможете ответить “да” на “описания

документированы и доступны команде и стейкхолдерам”, если вы не будете знать, какие именно нужны описания).

- контрольные вопросы — это не всё, что нужно для успешного выполнения проекта. Это просто “банальности, про которые иногда забывают”. Только отвечая на контрольные вопросы, проекта не выполнишь.

### Карточки состояний

Обычно контрольные вопросы оформляются в виде карточек состояний альф (да, карточек из плотной бумаги — типа игральных карт, cards). Вот пример оформления такой карточки для альфы “стейкхолдеры”, состояния “признаны” (первое состояние из шести):




Обратите внимание, что контрольные вопросы на карточках задаются не в терминах рабочих продуктов, а в терминах конечных результатов деятельности, в терминах дисциплины: напомним, что состояние одной альфы может свидетельствоваться десятком рабочих продуктов, но именно альфы используются для понимания ситуации. В этом-то и прелесть использования альф-из-дисциплины: они экономят мышление в ситуации с разнообразными рабочими продуктами и использующимися при их подготовке инструментами (компьютерными программами, бумажными формами и т.д.).

С другой стороны, команда решает (а иногда за команду решают ГОСТы, внутренние регламенты/стандарты организации, распоряжения, условия договора и т.д. — когда у команды нет права определять свой способ работы), какие рабочие продукты и в какой степени детальности их разработки нужно использовать для ответов на утверждения карточек альф.

Ответы на вопросы карточек предполагают не только устное согласие членов команды с утверждением (в маленьких командах и небольших проектах это вполне приемлемо), но и возможность демонстрации рабочих продуктов в любой момент — чтобы прояснить или удостоверить ответ. “Доверяй, но проверяй” — это и есть системная инженерия. В авиационных чеклистах при ответе на вопрос, достаточно ли топлива, нужно не просто отвечать “да” по “внутренней уверенности”, а действительно убедиться в достаточности топлива, взглянув на прибор. Состояние

рабочих продуктов должно соответствовать ответам на вопросы, и это должно быть демонстрируемо.

### Когда заводить подальфы

Очень часто невозможно ясно ответить на контрольный вопрос основной альфы (а иногда и подальфы):

- Непонятно, что происходит в проекте — слишком много разных объектов контроля упаковано в одном "да" или "нет", слишком много нужно принять во внимание
- Непонятно, что делать в каком-то особом случае, из-за которого задерживается ответ на весь вопрос
- Ответ получается слишком общий, не учитывает важных деталей

В этом случае правильным является определение подальфы, продвижение которой по состояниям ведёт к продвижению по состояниям основной альфы — и так по всей цепочке зависимости подальф. Так, "определение системы" продвигается по своим состояниям по мере продвижения по состояниям требований, архитектуры, неархитектурных инженерных решений. "Стейкхолдеры" продвигаются по состояниям каждый раз, когда продвигается по состояниям одиночный "стейкхолдер". Работа продвигается с каждым продвижением в составлении планов, с каждым выполняемым делом.

Определение подальфы состоит из следующих шагов:

- Нужно определить, что она из себя представляет (из какой она дисциплины — это было бы лучше всего, но инженеры могут создавать альфы и "по наитию", не на основе теории, а на основе эвристик. Главное, это не путать альфу и рабочий продукт, свидетельствующий её состояние). Не лишне вернуться и почитать подробней раздел про альфы и рабочие продукты.
- Нужно определить, подальфой каких альф (или подальф) она будет — в том числе сразу нескольких (так, "член команды" это подальфа альф "команды" и "стейкхолдеры" одновременно).
- Нужно определить, какие основные состояния будут у этой подальфы, определяющие её жизненный цикл.
- Сформулировать контрольные вопросы к каждому состоянию
- Оформить результат в виде набора карточек
- Определить, какие рабочие продукты должны будут свидетельствовать прохождение состояний этой новой альфы, если это описания, то какие практики описаний будут использованы для этих продуктов и какие инструменты будут поддерживать эти практики описаний.
- Если какие-то рабочие продукты окажутся отсутствующими, то разработать их на основе библиотечных (т.е. уже известных) практик описания. Если не нашлось подходящей практики описания, то создать её!

### Карточные игры

С карточками проводится довольно много самых разных "карточных игр". В системной инженерии "игрой" называется какое-то ролевое поведение, проходящее по строгим правилам в виде "ходов". Игры весьма распространены в agile видах

жизненного цикла. С карточками альф может проводиться множество самых разных игр:

- Покер: члены команды имеют собственный набор карточек и выкладывают на стол ту карточку рубашкой вверх, которую они считают наиболее соответствующей реальному положению дел в проекте. Потом все одновременно переворачивают карточки — если карточки-состояния оказались одинаковыми, то всё ОК. Если неодинаковыми (бывает много чаще), то это повод выяснить, почему у разных членов команды разное мнение — и тем самым синхронизировать понимание ситуации.
- Планирование работы: слева выкладываются карточки, которые уже пройдены, а справа — ещё не пройденные карточки. Выясняются проблемы, которые не дают возможность "чекнуть" следующую карточку, и эти проблемы (или сразу по обсуждении — задачи, следующие из этих проблем) документируются в форме дел (issues) в системе ведения дел.
- Больше игр см. тут: http://www.ivarjacobson.com/alphastatecards/

Работа с карточками связана с обнаружением проблем и планированием задач для их решения. Для учёта проблем, переходящих в задачи, а после их решения становящихся уведомлениями о решении проблем, используются специальные системы отслеживания дел — issue trackers. Когда-то такие системы появились для учёта обнаруживающихся в проекте ошибок (bugs), но довольно скоро они были обобщены до отслеживания "вопросов" (issues, мы будем переводить их как "дела"). Эти системы ведения дел обычно тесно связаны с системами поддержки версионности (управления конфигурацией).

Каждый раз, когда обнаруживаются проблемы, мешающие честно сказать "да" для ответа на вопросы карточек, эти проблемы заносятся в issue tracker и далее отслеживается решение этих проблем ("закрытие дел").

### *Контрольные вопросы инженерного проекта*

#### Карточки основных альф инженерного проекта

Далее будут приведены карточки основных альф инженерного проекта, но не в виде карточек, а в виде набора таблиц — кроме контрольных вопросов в этих таблицах будут приведены примеры видов рабочих продуктов, которые можно использовать для свидетельствования этих состояний альф (в больших рабочих продуктах может быть указана только их часть, иногда даже отдельное поле), и прокомментирована практика описания этих рабочих продуктов (помним, что для любого описания есть своя практика описания — каждое view имеет свой viewpoint).

Нужно помнить, что все эти замечания про рабочие продукты и практики их описания даны только "например". Сами контрольные вопросы довольно устойчивы к разнообразию инженерных проектов и используемым в них самым разным практикам достижения проектных целей, но вот рабочие продукты и инструменты, а также методы описания для одних и тех же описаний могут быть крайне разнообразны. Кроме того, в разных проектах могут быть самые разные требования к документированию: в некоторых проектах для свидетельствования состояния альфы нужно просто получить согласие всех членов команды на совещании, а в некоторых проектах придётся предъявлять документированные обоснования, правильность которых будет контролироваться внешними по отношению к команде инспекторами ("независимая экспертиза"). Жизнь сложна и разнообразна, нельзя

принимать карточки и их контрольные вопросы как догму, и ещё меньше догмой являются способы их использования, рабочие продукты и практики их описания, а также инструменты для работы с рабочими продуктами.

В большинстве случаев вместо длинной фразы "метод описания, принятый в выбранной для проекта практике XXX" приведена более короткая (хотя и менее точная фраза) "практика XXX". Для различных опросов фраза "методика опросов и формат отображения результата" в таблицах не прописана.

Ну, и нужно понимать, что карточки альф инженерного проекта даны в их самом универсальном виде — но в зависимости от паттерна жизненного цикла (http://csse.usc.edu/csse/TECHRPTS/2009/usc-csse-2009-502/usc-csse-2009-502.pdf — "купи готовое" … "акцент на сервисах") и жёсткости регулирования (например, военные разработки, или разработки атомной энергетики, или даже разработки в строительстве с его лицензированием и детальными нормативными документами отличаются детальной прописанностью многих рабочих процедур), так что без адаптации их к условиям конкретного проекта использовать их может оказаться затруднительно.

В любом случае, в маленьких проектах значительная часть решений будет приниматься "с голоса", и не будет требовать письменного оформления в виде каких-то актов, протоколов, списков и т.д. Но в больших проектах можно будет встретить огромное количество рабочих продуктов, свидетельствующих состояния альф — настоящую "инженерную бюрократию", да плюс ещё и "менеджерскую бюрократию", связанную с ресурсным обеспечением и операционным управлением проекта.

Так что в нижеприведённых таблицах обращайте прежде всего внимание на суть задаваемых вопросов, а примеры рабочих продуктов и практик их описания даны только для того, чтобы проиллюстрировать использование практики контрольных вопросов для "тяжёлых" разработок в больших инженерных предприятиях и сильно зарегулированных отраслях. Не забывайте, что нынешний тренд — это уменьшение "бюрократии", но без ущерба для дела. Поэтому контрольные вопросы будут оставаться, но вот письменные развёрнутые доказательства со специализированными рабочими продуктами и хитрыми методиками их разработки в ответ на каждый вопрос — вряд ли, ибо цель вопросов не "отчитаться за проделанную работу", а просто обратить внимание команды на текущее положение дел, не дать забыть какую-нибудь банальность типа известности источника финансирования или наличия механизма управления конфигурацией. Трудно поверить, но в пылу авралов забывается и такое: карточки с контрольными вопросами помогают не упустить леса за деревьями.

### Стейкхолдеры

Основные практики работы со стейкхолдерами — это обеспечение того, чтобы представители групп стейкхолдеров/исполнители ролей стейкхолдеров активно участвовали в разработке, т.е. практики лидерства и коллаборации.

Стейкхолдеров обычно много (для удобного их группирования часто используют луковичную диаграмму — http://businessanalystlearnings.com/ba-techniques/2013/1/22/how-to-draw-a-stakeholder-onion-diagram), и все они обычно находятся в разном состоянии в какой-то момент времени. Поэтому хорошей практикой является определение подальфы "стейкхолдер", и далее работа с отдельными экземплярами этой альфы (понимая, что продвижение по состояниям

каждого из стейкхолдеров ведёт/drive к продвижению по состояниям всей альфы "стейкхолдеры").

Помним также, что "подрядчик" и "член команды" — это тоже подальфы "стейкхолдеров".

**Признаны**

Стейкхолдеры были определены.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Все различные группы стейкхолдеров, которые на данный момент, или в будущем будут затронуты разработкой и функционированием инженерной системы, определены. | "Длинный список" стейкхолдеров (роли) | Таблица в Excel/Word с ролями и описаниями ролей (или описания ролей даны где-то в отдельном документе) Луковичная диаграмма |
| Есть соглашение по группам стейкхолдеров, которые будут представлены. Как минимум, должны приниматься в расчёт группы стейкхолдеров, которые финансируют, используют, поддерживают и обслуживают систему. | Завизированный командой "короткий список" ролей (в CRM-системе, таблице стейкхолдеров, и т.д.) | Отметка в "длинном списке" для ролей о представлении и ссылка на соглашение (протокол, распоряжение и т.д.) |
| Ответственности представителей стейкхолдеров были определены. | Запись об ответственности (в CRM-системе, таблице стейкхолдеров и т.д.) | Запись об ответственности (какие решения он полномочен принимать, в какие сроки реагировать, обязательства по присутствию на совещаниях, согласования документов и т.д.) |

**Представлены**

Механизмы вовлечения стейкхолдеров согласованы и назначены представители стейкхолдеров

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Представители стейкхолдеров согласились выполнять свои обязанности | Отметка о согласии (в CRM-системе, таблице стейкхолдеров и т.д.) | Ссылка на документ (письмо, протокол совещания и т.д.) или протоколирование факта (устное согласие) |
| Представители стейкхолдеров | Отметка об уполномоченности | Ссылка на уполномачивающий |

| | | |
|---|---|---|
| уполномочены выполнять свои обязанности | | документ (приказ, назначение и т.д.) или обоснование достаточности полномочий (уже имеющиеся полномочия "первого лица"), приказ на создание рабочей группы с включением внешних представителей стейкхолдеров "по согласованию" |
| Подход к обеспечению сотрудничества среди представителей стейкхолдеров был согласован | Завизированный список мер по обеспечению сотрудничества (виды совещаний — например, очные или селекторные, графики обмена документами, использование issue tracker, допуск в информационные системы проекта, участие в списках рассылки и т.д.) | VISI для госстроительства в Дании<br>Лучшие практики коллаборации, сотрудничества, взаимодействия |
| Представители стейкхолдеров поддерживают и уважают технологию работы команды | Запись в протоколе совещания, на котором представлена технология работы команды, согласованный с представителями стейкхолдерами вид жизненного цикла со ссылкой на доступное представителям стейкхолдеров описание технологий.<br>Виза стейкхолдеров на описании технологии то есть на перечислении практик, вида жизненного цикла, документах архитектуры предпринятия, наборах технологических стандартов организации и т.д.<br>Результаты опроса стейкхолдеров. | |

**Вовлечены**

Представители стейкхолдеров активно вовлечены в работу и выполняют свои обязанности.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Представители стейкхолдеров помогают команде в соответствии со своими обязанностями. | Результаты опроса команды.<br>Метрики прохождения дел в issue tracker по помощи команде со стороны стейкхолдеров. | Дело в issue tracker, исполнителем которого можно назначить стейкхолдера |
| Представители стейкхолдеров обеспечивают обратную связь и принимают участие в принятии решений своевременно | Нет "висящих" дел по проблемам несоответствия оценки представителя стейкхолдера и его группы (обратная связь) и нет просроченных согласований со стейкхолдерами (предоставление подписанных документов).<br>Результаты опроса команды по адекватности реакции стейкхолдеров на их запросы. | Отслеживание дат назначения дел и контрольных сроков дел в issue tracker |
| Представители стейкхолдеров быстро сообщают изменения, которые имеют значение для их групп стейкхолдеров | Опрос команды по "непрохождению сигнала" от группы стейкхолдеров через представителя к команде (отсутствие инициативы стейкхолдера по сообщению важных событий). | Дело в issue tracker, которое может инициативно завести представитель стейкхолдеров. |

**В согласии**

Представители стейкхолдеров в согласии.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Представители стейкхолдеров согласились с минимальными ожиданиями для последующего разворачивания новой системы. | Завизированный представителями стейкхолдеров раздел технических требований в техническом задании.<br>Завизированный представителями стейкхолдеров минимальный набор критериев приёмки целевой системы.<br>Визы представителей стейкхолдеров на рабочих | На рабочих продуктах есть места для виз представителей стейкхолдеров.<br>Практика инженерии требований |

| | | |
|---|---|---|
| | продуктах, содержащих “требования к системе”. | |
| Представители стейкхолдеров удовлетворены своей вовлечённостью в работу. | Результаты опроса стейкхолдеров (ответ на вопрос “удовлетворены ли вы вашим участием в работах проекта”) | |
| Представители стейкхолдеров согласны, что их вклад в работу ценится командой и учитывается в работе с уважением. | Строчка в протоколе совещания<br>Результаты опроса стейкхолдеров | |
| Члены команды согласны, что их вклад в работу ценится представителями стейкхолдеров и учитывается в работе с уважением. | Строчка в протоколе совещания.<br>Результаты опроса команды | |
| Представители стейкхолдеров согласны с тем, как их различные приоритеты и точки зрения балансируются для обеспечения ясного руководства для команды. | Строчка в протоколе совещания.<br>Результаты опроса стейкхолдеров. | |

**Удовлетворены для разворачивания**

Минимальные ожидания представителей стейкхолдеров были достигнуты.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Представители стейкхолдеров обеспечивают обратную связь с точки зрения их групп стейкхолдеров. | Доверенность.<br>Приказ руководства представителя стейкхолдера.<br>Результаты опроса команды (отсутствие возражений) в случае “внутреннего представителя”. | Типовые формулировки доверенности. Типовые формулировки приказа. |
| Представители стейкхолдеров подтверждают, что система готова для разворачивания. | Визы стейкхолдеров на акте готовности к разворачиванию (акт приёмки-сдачи, акт передачи в эксплуатацию, акт о начале эксплуатации).<br>Отсутствие незакрытых дел, заведённых стейкхолдерами как | Есть места для виз стейкхолдеров на акте готовности к разворачиванию.<br>Возможность заведения дел в issue tracker представителями стейкхолдеров.<br>Визы стейкхолдеров на актах приёмо-сдаточных |

| | | |
|---|---|---|
| | условие для готовности к разворачиванию (отсутствие неотвеченных замечаний). | испытаний. |

**Удовлетворены в использовании**

Система удовлетворяет или превышает минимальные ожидания стейкхолдеров

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Стейкхолдеры используют новую систему и предоставляют обратную связь об их опыте. | Эксплуатационные отчёты | Формат исторических данных (показания датчиков, журналы использования, журналы ремонтов и техобслуживания) |
| Стейкхолдеры подтверждают, что новая система соответствует их ожиданиям | Протокол об отсутствии рекламаций.<br>Результаты опросов | |

Возможности

Основа работы с возможностями — это предпринимательство, которое принципиально неформализуемо, ибо связано как-то с оценками будущего. В современном мире предпринимательские решения слабо поддаются формализации. Это относится и к таким решениям, как принятие заказа к исполнению в зависимости от текущей загрузки ресурсов — принятие решения на основе теории предельной (marginal/маржевой/граничной) полезности (utility) и маржинальных цен, или инвестиционные решения в проработку возможностей по заключению контракта или выходу на рынок. Но в силу сложности предпринимательской деятельности и коллективного её характера появилось довольно много практик формализации и документирования промежуточных для принятия предпринимательских (инвестиционных, стратегических, маркетинговых) решений.

Пожалуй, альфа возможности — это самая сложная для работы альфа. По сложности работы с ней может сравниться только альфа команды. С другой стороны, инженер может стать великим предпринимателем и великим лидером. Но отнюдь не каждый великий предприниматель и великий лидер может стать инженером, так что к оценкам относительной сложности работы с разными альфами нужно относиться очень осторожно: что просто для одного человека, может быть запредельно трудно и даже невозможно для другого, и наборот.

Среди подальф возможностей нужно особо указать “бюджет”, с которым работают практики бюджетирования (включая такие, как beyond budgeting), а также “потребности” (stakeholder needs), с которыми работают практики анализа потребностей (user needs analysis, целеориентированная инженерия требований и т.д.).

**Определена**

Коммерческая, общественная или инвестиционная возможность, которая могла бы быть адресована инженерным решением, определена.

| Контрольные вопросы | Пример рабочих продуктов | Пример практик описания |
|---|---|---|

| | (views) | (viewpoints) |
|---|---|---|
| Идея по способу улучшения текущих технологий работы, увеличения рыночной доли или по применению новой или инновационной инженерной системы была определена. | Описание целей проекта | Раздел в заявлении на открытие проекта |
| Как минимум один из стейкхолдеров желает сделать инвестицию в более подробное понимание возможности и пользы, связанной с адресацией этой возможности. | Подпись имеющего ресурсы стейкхолдера на заявлении на открытии проекта | Место для подписи в заявлении об открытии проекта |
| Другие стейкхолдеры, для которых это общая возможность, определены. | Список стейкхолдеров | Формат списка стейкхолдеров |

**Нужно решение**

Потребность в инженерном решении была подтверждена.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Стейкхолдеры для возможности и предложенное решение были определены. | Список стейкхолдеров | Формат списка стейкхолдеров |
| Потребности стейкхолдеров, которые порождают возможность, были установлены. | Документированные нужды стейкхолдеров | Формат описания нужд стейкхолдеров (практика user needs analysis) |
| Любые связанные с возможностью проблемы и их корневые причины были определены. | Список рисков | Практика управления рисками |
| Было подтверждено, что инженерное решение нужно. | Подпись руководителя на решении (приказе, project charter, других формах инициирования проекта) | Практика предпринимательства |
| По меньшей мере одно инженерное решение было предложено. | Предложена верхнеуровневая архитектура решения | Формат представления эскизной архитектуры и требования к ней для данного класса решений |

**Польза установлена**

Польза успешного решения была установлена

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|

| | | |
|---|---|---|
| Польза реализации возможности была определена количественно либо в абсолютных значениях, либо в единицах дохода или экономии за период (например, за год). | Технико-экономическое обоснование необходимости системы для клиента | Практики технико-экономических обоснований, стратегирования, оценки ROI и т.д. |
| Влияние решения на стейкхолдеров понятно. | Описание удовлетворения needs | Таблица, в которой описаны колонки needs и влияния системы на них |
| Польза, которую инженерная система предлагает стейкхолдерам, которые финансируют и используют систему, понятна. | User needs<br>Технико-экономическое обоснование | Практика "User needs analysis"<br>Практика технико-экономических обоснований (оценки стоимости). |
| Критерии успеха, по которым будет приниматься решение о разворачивании системы, ясны. | Меморандум о целях проекта с визами стейкхолдеров | Ключевые характеристики технического решения |
| Желаемые результаты, требуемые от решения, ясны и определены количественно. | Архитектурные требования (основные требования, влияющие на архитектуру) | Формат представления требований (в части архитектурных требований) |

**Жизнеспособна**

Согласовано, что решение может быть произведено достаточно быстро и дёшево, чтобы успешно реализовать возможность.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Решение обрисовано в общих чертах. | Архитектурное описание | Архитектурные практики |
| Есть признаки, что решение может быть разработано и развёрнуто в текущих ограничениях. | Экспертное заключение по архитектуре | Архитектурные практики |
| Риски, связанные с решением, приемлемы и управляемы. | Таблица рисков и способа ответов на них | Практики управления рисками |
| Грубая оценка цены решения меньше, чем ожидаемая польза от реализации возможности. | Бюджет проекта в сравнении с технико-экономическим обоснованием | Практики бюджетирования и технико-экономического обоснования |
| Причины для разработки инженерного решения понимаются всеми членами команды. | Проведение kick-off meeting | Практики презентации и контроля понимания |
| Ясно, что реализация | Экспертное заключение | Практика feasibilities study |

| | | |
|---|---|---|
| возможности жизнеспособна. | | |

**Реализована**

Решение, которое произведено, демонстрирует реализацию возможности.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Готовая к использованию система, которая демонстрирует реализацию возможности, доступна. | Готовая система | Конфигурационная ведомость |
| Стейкхолдеры согласны, что доступное решение заслуживает разворачивания. | Акт о готовности к опытной эксплуатации<br>Приказ о переходе к эксплуатации | |
| Стейкхолдеры удовлетворены тем, как разработанное решение адресует возможность. | Подписание акта приёмки-сдачи<br>Результаты опроса | |

**Извлекается выгода**

Эксплуатация или продажа решения создаёт осязаемые выгоды.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Решение начало извлекать выгоды для стейкхолдеров. | Акт о начале эксплуатации | |
| Профиль возврата инвестиций по меньшей мере так хорош, как ожидалось. | Результат опроса стейкхолдеров-пользователей<br>Сверка бюджета инженерной команды | |

Определение системы

Определение системы в целом для инженерного проекта — это самый-самый верхний уровень детализации проекта, больше пригодный для общения с менеджерами, нежели для собственно инженерии.

С определением системы лучше работать, разбивая его на различные подальфы: как соответствующие отдельным частям-модулям (ибо речь идёт об управлении разработкой), так и соответствующие различным видам описаний: подальфы требований, архитектуры, неархитектурной части проекта.

Определение системы, даже если рассматривать его через подальфы, также крайне разношёрстно в части практик описания (так только архитектурное описание в стандарте DoDAF имеет 28 видов групп описаний/viewpoint. Крайне разнообразы также и рабочие продукты — проектная документация занимает много томов, инженерные базы данных имеют крайне сложные модели данных). Поэтому примеры практик работы с альфой определения системы носят очень условный

характер.

**Замыслено**

Ясно, каково будет определение системы.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Ясно, что будет считаться успехом для новой системы. | User needs | Практика Needs analysis |
| Методы описания системы согласованы. | Список документов и стандартов по их подготовке | Указание на ЕСКД, СПДС, 3D моделирование |
| Способ согласования описаний со стейкхолдерами согласован. | Завизированный меморандум по согласованиям | Указывается способ предоставления доступа к рабочим продуктам, формат и способ запросов на изменения (указания проблем), графики и сроки предоставления обратной связи, необходимые визы и т.д. |
| Механизмы управления конфигурацией описаний согласованы. | Завизированный командой меморандум по управлению конфигурацией | Указываются система хранения версий, система управления изменениями, регламенты по работе с ними, конкретные папки или базы данных, заведённые для проекта |

**Непротиворечиво**

Определение системы создано и непротиворечиво.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Описания документированы и доступны команде и стейкхолдерам. | База данных в системе управления конфигурацией, все члены команды и стейкхолдеры имеют доступ к этой базе данных | Практика управления конфигурацией |
| Происхождение описаний ясно. | Мета-информация об авторе (при этом каждый автор имеет свой эккаунт) | Эккаунт в системе управления конфигурацией для каждого члена команды и стейкхолдера |
| Описания проверяются. | Наличие отметок о проверках, а не только об окончании работ (возможно, реализуется как статус issue в issue tracker) | Роли исполнителя и принимающего работу разведены |

| | | |
|---|---|---|
| Противоречивые описания идентифицированы и ими занимаются. | Issue, порождаемые по обнаруженным коллизиям, эти issue назначаются на исполнителей и по ним проводится работа. | Практика проверок (review, инструментальные проверки, сборки конфигурации и т.д.)<br>Практика управления изменениями |
| Команда понимает описания и соглашается их воплотить. | Утверждённое архитектурное описание<br>Результаты опроса | Практика архитектурного проектирования<br>Практики MBSE |
| Система, соответствующая описаниям, принимается стейкхолдерами как заслуживающая воплощения. | Решение о начале производства | |

**Используется для изготовления**

Определение системы используется для изготовления системы.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Подготовлено достаточное количество описаний системы, чтобы начать изготавливать систему. | Рабочая документация (неархитектурные решения) | Стандарты конструкторской и проектной документации |
| Технологии изготовления определены. | Технологическая документация | Стандарты технологической документации |
| Часть команды, изготавливающая систему, признаёт описания достаточными для изготовления системы. | Акт об успешном hand over<br>Уведомление о начале производства<br>Визы технологов на проектной документации/моделях | Практики hand-over (передачи всей необходимой информации на очередную стадию жизненного цикла) |
| Возникающие при изготовлении проблемы приводят к переработке и актуализации определения системы. | Запросы на изменения, проходящие обработку | Практика управления изменениями |

**Используется для проверки воплощения**

Определение системы используется для проведения тестов и испытаний.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Нет частей определения системы, без которых проверки невозможны. | Проектная документация/модели<br>Конфигурационная ведомость | Архитектурное проектирование<br>Управление конфигурацией |
| Проверки, критерии их успешности и способ их проведения определены. | Программа испытаний | Практики проверки и приёмки (verification and validation, V&V) |
| Стейкхолдеры согласны с | Виза стейкхолдеров на | Практика военной приёмки |

| | | |
|---|---|---|
| объемом проверок. | программе испытаний | |

**Используется для эксплуатации**

Определение системы используется стейкхолдерами для её эксплуатации.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Определение системы используется для сбора информации о состоянии эксплуатируемого воплощения системы. | Формат журнала регистрации замеров датчиков, проведения обслуживания и т.д. | Модели данных исторической информации (стандарты O&M MIMOSA) |
| Определение системы наряду с информацией о состоянии эксплуатируемой системы используется для принятия решений о техобслуживании, ремонтах, модернизации. | Модели, позволяющие определить состояние системы по исторической информации | Модели надёжности |

**Используется для вывода из эксплуатации**

Определение системы используется для ликвидации и/или переработки системы.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Определение системы используется для определения момента вывода из эксплуатации или принятии решения о продлении эксплуатации. | Модели надёжности<br>Технико-экономические модели | Практики расчётов надёжности и технико-экономического моделирования<br>Практики продления сроков службы системы |
| Определение системы демонстрирует отсутствие вредных эффектов (например, загрязнения окружающей среды) при выводе системы из эксплуатации. | Обоснование безопасности при выводе из эксплуатации | Практики инженерных обоснований (assurance case), в том числе практики обоснований соответствия стандартам (compliance). |
| Определение системы используется для планирования и проведения работ по ликвидации и/или переработке воплощения системы. | Руководство по выводу из эксплуатации | Практики вывода из эксплуатации по ISO 15288, отраслевые практики вывода из эксплуатации |

Воплощение системы

Получить успешное (удовлетворяющее стейкхолдеров) воплощение системы — это и есть главная цель инженерного проекта, даже определение системы нужно

главным образом как средство получить воплощение системы, а не само по себе. Традиционно практики системной инженерии работают с воплощением системы больше на конечных состояниях этой альфы, а именно в части практик проверки и приёмки. А ещё важно при воплощении системы управлять конфигурацией: чтобы всё запроектированное было закуплено/изготовлено, и именно оно попало в состав целевой системы, и именно целевая система затем была налажена, проверена-принята и пошла в эксплуатацию.

Обращаем внимание на то, что состояния альф отнюдь не заканчиваются сдачей в эксплуатацию: жизненный цикл системы всегда полный — до самого её вывода из эксплуатации (включая уничтожение или переработку). И карточки с контрольными вопросами не дают об этом забыть.

Подальфы определения системы чаще всего — это подальфы отдельных модулей, требующих особого контроля.

**В виде сырья**
Материалы для воплощения системы наличествуют и готовы к изготовлению частей.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Материалы для воплощения системы наличествуют и позволяют создать детали с нужными характеристиками. | Сводный список материалов и комплектующих (библиотек в случае программных систем) с оценками их доступности | Практики цепочки поставок (supply chain), форматы data sheets и опросных листов с характеристиками материалов и оборудования |
| Оборудование для переработки материалов в детали наличествует. | Список оборудования/инструментов с оценками доступности | Архитектура предприятия |
| График производства и логистики частей системы согласован. | Визы на графике производства и логистических графиков | Практики планирования проектов |
| Возможны работы по изготовлению частей. | Технико-коммерческие предложения от подрядчиков | Типовые форматы технико-экономических предложений |

**В виде частей (parts)**
Части системы созданы и готовы к интеграции.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Части системы созданы и/или закуплены и проверены. | Части системы<br>Накладные<br>Акты входных испытаний | Управление конфигурацией |
| График интеграции (сборки, монтажа, строительства) из частей согласован. | Визы на графике интеграции | Проектное управление |
| Возможны работы по интеграции (сборке, монтажу, строительству). | Уведомление о начале сборки, монтажа, строительства | |

**Демонстрируемо**

Система собрана из её частей и готова к проверке.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Система может быть опробована в её отдельных функциях и её ключевые характеристики могут быть измерены. | Наличие собранной системы (акт) | |
| Ключевые характеристики системы могут быть продемонстрированы. | Оборудование для испытаний, наборы тестов | Практики проверки и приёмки |
| Критические интерфейсы были продемонстрированы. | Акты испытаний интерфейсов | |
| Интеграция с другими существующими системами была продемонстрирована. | Акты испытаний интерфейсов | |
| Необходимые стейкхолдеры согласны, что систему нужно проверять. | Решение о проведении испытаний | |

**Готово**

Система (как целое) была принята для эксплуатации её в операционном окружении.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Функциональность, обеспечиваемая системой, протестирована. | Протоколы испытаний | Практики проверки и приёмки |
| Уровни дефектов приемлемы для стейкхолдеров. | Отсутствие протокола о доделках | |
| Установочная и другая пользовательская документация доступна. | Установочная и пользовательская документация: указание на репозиторий | Управление конфигурацией |
| Представители стейкхолдеров принимают систему, как удовлетворяющую своему назначению. | Акт приёмки-сдачи | |
| Состав передаваемой стейкхолдерам системы известен. | Виза стейкхолдеров на конфигурационной ведомости | Управление конфигурацией |
| Представители стейкхолдеров хотят принять систему в | Приказ стейкхолдера о начале эксплуатации системы | |

| | | |
|---|---|---|
| эксплуатацию. | | |
| Эксплуатационная поддержка наличествует. | Контакты службы эксплуатационной поддержки, виза службы эксплуатационной поддержки о принятии на обслуживание | Организация сервиса (ITIL в случае программных систем). |

**Эксплуатируется**

Система эксплуатируется в её операционном окружении.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Система сделана доступной стейкхолдерам, которые намерены её использовать. | Наличие пользователей | |
| Есть как минимум один пример полностью работающей системы. | Работающая система | |
| Система полностью поддерживается на согласованном уровне сервиса. | Соглашение SLA о поддержке системы<br>Журналы мониторинга уровня сервиса | Организация сервиса (ITIL в случае программных систем) — хелпдеск, отслеживание инцидентов и т.д. |

**Выведено из эксплуатации**

Воплощение системы больше не поддерживается, система ликвидирована и/или переработана.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Воплощение системы было заменено или прекращено в использовании. | Акт о выводе из эксплуатации | |
| Система больше не поддерживается. | Приказ об окончании поддержки<br>Акт о закрытии договора поддержки<br>Уведомление об окончании поддержки | |
| Нет «официальных» стейкхолдеров, которые до сих пор используют систему. | Отсутствие стейкхолдеров | |
| Доработки /доделки системы больше не будут производиться. | Приказ об окончании работ | |
| Все материальные компоненты системы либо повторно используются, либо надлежащим образом | Акты о ликвидации<br>Акты о передаче на баланс других проектов | Практика вывода из эксплуатации |

| | | |
|---|---|---|
| ликвидированы. | | |

Команда

Основные трудности в том, что практики работы с командой крайне неформализованы, имеют огромное количество вариаций в связи с личностным стилем руководителя. Они главным образом определяются дисциплиной “лидерство” (leadership), которая сама по себе крайне неформальна (хотя часть этих практик определяется более формализованными практиками human resources management и менее распространёнными talent management). В последнее время практики лидерства и командообразования для технических проектов изучаются в рамках развития методологий для agile видов жизненного цикла. Мы тут воздержимся от примеров практик, в которых определяются описания рабочих продуктов, используемых для работы с альфой команды — это совершенно особый предмет.

Подальфы команды — это “член команды” и “подрядчик”. Но могут быть и менее очевидные альфы, например, “сотрудничество” (которого вначале нет, а потом оно должно появиться) и “ресурсы” (которых вначале может не быть, но которые затем должны появиться): это всё разные аспекты “команды”, с которыми работают разные люди предприятия.

**Намечена**

Миссия команды ясна и знания о том, как растить команду, наличествуют.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Миссия команды определена в терминах возможностей и результатов. | Рабочие продукты возможностей | |
| Ограничения на работу команды известны. | Список ограничений (что команда не может устанавливать сама) | |
| Механизмы для роста команды наличествуют. | Регламент добавления людей в команду | |
| Состав команды определён. | Утверждённый список ролей команды | |
| Все ограничения, определяющие где и как будет выполняться работа, определены. | Распоряжение по рабочим местам членов команды, местам проведения совещаний, предоставлению доступа к инструментам и т.д. | |
| Обязанности команды обрисованы в общих чертах. | Project charter | |
| Уровень принятых командой обязательств ясен. | Kick-off meeting | |
| Требуемые компетенции определены. | Список компетенций для каждой роли в команде | |

| | | |
|---|---|---|
| Размер команды определён. | Количество людей для каждой роли | |
| Правила контроля за деятельностью определены. | Приказ: кому подчиняется команда | |
| Форма управления выбрана. | Регламент, определяющий управление в команде | |

**Сформирована**

Команда была пополнена достаточным количеством людей с принятыми обязательствами, чтобы начать миссию.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Индивидуальные обязанности понимаются. | Регламенты ролей<br>Распределение обязанностей на kick-off meeting | |
| Было набрано достаточное число членов команды, чтобы работа продвигалась. | Список команды (включая исполнителей) | |
| Каждый член команды понимает, как команда организована, и какая у него индивидуальная роль. | Kick-off meeting<br>Репозиторий документов проекта<br>Приказ о назначении членов команды | |
| Все члены команды понимают, как выполнять их работу. | Аттестация<br>Технические совещания | |
| Все члены команды встретились (возможно, виртуально) и начинают узнавать друг друга. | Kick-off meeting<br>Социальные метрики | |
| Члены команды понимают их обязанности и как они увязаны с их компетенциями. | Kick-off meeting<br>Собеседования | |
| Члены команды принимают работу. | Назначение issues<br>Назначение работ по плану | |
| Любые внешние смежники (организации, команды и индивиды) определены. | Список стейкхолдеров-подрядчиков | |
| Механизмы общения в команде определены | Меморандум о коммуникации | |
| Каждый член команды принял обязательство работать в команде, как | Kick-off meeting | |

| | | |
|---|---|---|
| это определено. | | |

### Сотрудничает

Члены команды работают вместе как одно подразделение.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Команда работает как одно сплочённое подразделение. | Значения метрик производительности | |
| Общение в команде открытое и честное. | Результаты опроса | |
| Команда сфокусирована на достижение миссии команды. | Результаты опроса | |
| Члены команды знают друг друга. | Результаты опроса | |

### Производит

Команда работает результативно и эффективно.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Команда систематически выполняет обязательства. | Значения метрик производительности | |
| Команда непрерывно адаптируется к изменяющемуся контексту. | Изменения технологий | |
| Команда определяет и адресует проблемы без внешней помощи. | Оценка руководства | |
| Прогресс в результатах достигается с минимальным необходимым возвращением к сделанному и переделками. | Метрика переделок | |
| Работа впустую и причины для работы впустую постоянно устраняются. | Метрики производительности и изменения технологий | |

### Распущена

Команда больше не ответственна за выполнение своей миссии.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Обязанности команды были переданы или прекращены. | Приказ об окончании работ | |
| Члены команды доступны | Результаты опроса | |

| | | |
|---|---|---|
| для назначения в другие команды. | | |
| Командой не предпринимается дальше никаких усилий для завершения миссии. | Результаты опроса | |

Работа

Нужно учитывать, что в инженерной работе используется одновременно как минимум несколько различных дисциплин управления работами:

- Управление процессами (как повторяющейся деятельностью). Все регламенты — они именно про "процессы"
- Управление проектами, с их парадигмой предварительного календарного планирования и контролем выполнения графика
- Управление делами (case management), относительно новая дисциплина, использующаяся в управлении плохо планирующимися деятельностями (лечение больных, проектирование принципиально новых систем, исследования).

Каждая инженерная компания по-разному делает упор на эти разные дисциплины в зависимости от вида жизненного цикла (паттерна разработки) и управленческих традиций, разделяемых менеджерами самой команды и топ-менеджерами.

Наиболее очевидные подальфы у "работ" будут "план-график" и "задача".

**Инициирована**

Работа была запрошена.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Результат, требуемый от инициированной работы, ясен. | Рабочие продукты возможностей<br>Директивный график | Практики инженерии требований<br>Практики проектного управления |
| Любые ограничения на выполнение работы ясно определены. | Оценка свободных ресурсов | В том числе теория ограничений (в части нахождение ограничений) |
| Стейкхолдеры, которые будут финансировать работу, известны. | Список стейкхолдеров<br>Документы, подтверждающие согласие финансирования | Таблицы в Excel, отчёты CRM, меморандумы, докладные записки. |
| Инициатор работ известен. | Приказ на открытие работ | |
| Стейкхолдеры, которые будут принимать работу, известны. | Список стейкхолдеров | |
| Источник финансирования ясен. | Строка бюджета, оценка суммы | Внутренние практики бюджетирования. |
| Приоритет работы ясен. | Директивный график, критическая цепь | Теория ограничений в части приоритета проекта в контексте загрузки всех |

| | | |
|---|---|---|
| | | ресурсов и всех проектов предприятия. |

**Подготовлена**

Все предусловия для начала работы выполнены.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Обязательства приняты. | Подписанный договор | Типовой договор и типовое техническое задание к нему |
| Цена и потребные усилия оценены. | Соглашение о цене<br>Бюджет | Практики бюджетирования (в том числе практики beyond budgeting) |
| Доступность ресурсов понимается. | Данные управления ресурсами предприятия | Система проектного управления масштаба предприятия, ERP-система |
| Правила и процедуры контроля ясны. | Регламент управления проектами | В том числе использование карточек Essence |
| Риски понимаются. | Меморандум о рисках | Практики управления рисками |
| Критерии приёмки определены и согласованы с клиентом. | Подписанное техническое задание | Практики needs analysis, инженерии требований, проверки и приёмки |
| Работы разбиты достаточно для того, чтобы началась производительная работа. | Детальный план работ (графики 2-6 уровней)<br>Дела в issue tracker | Практики Use Case 2.0 (sliced Use Case), проектного управления, agile-планирования |
| Стейкхолдерами и командой задачи определены и приоритизированы. | Содержательное наполнение плана-графика | Практики системной инженерии, инженерного менеджмента |
| Правдоподобный план наличествует. | План-график | Проектное управление |
| Финансирование для начала работы наличествует. | Приказ о финансировании работ | |
| Команда или, как минимум, часть команды готова начать работу. | Результаты опроса | |
| Моменты интеграции и поставки определены. | Моменты в плане-графике | Практика гейтовой организации работ и директивного графика, определение вида жизненного цикла |

**Начата**

Работа происходит.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|

| | | |
|---|---|---|
| Работа по разработке начата. | Метрики производительности | Практика измерений |
| Прогресс работы отслеживается. | Система метрик производительности | Практика измерений |
| Работа разбита на единицы действий с ясными определениями того, что нужно сделать. | План-график<br>Дела в issue tracker | Управление проектами, управление делами |
| Члены команды принимают и выполняют задания. | Результаты опроса | |

**Под контролем**

Работа продвигается хорошо, риски под контролем, уровень производительности достаточен для достижения удовлетворительного результата.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Количество завершенных задач растёт. | Issue tracker<br>План-график с выполненными работами<br>График burn-out и другие графики метрик выполнения работы | Измерение производительности в проектном управлении, в agile методологиях |
| Незапланированная работа под контролем. | Issue tracker<br>Изменения в технологии<br>Учёт технического долга | Issue tracking<br>Постоянное совершенствование<br>Управление техническим долгом |
| Риски под контролем, их влияние, если они реализуются, и вероятность их реализации снижены до приемлемых уровней. | Апдейты списка рисков | Управление рисками |
| Оценки пересматриваются, чтобы отражать производительность команды. | Метрики производительности | Практики agile-работы, практики проектного управления |
| Доступны меры для показа продвижения и скорости работы. | Метрики производительности | Практики agile-работы, практики проектного управления |
| Переделки под контролем. | Issue tracker<br>Метрики переделок | Теория управления качеством (цикл Деминга) |
| Задачи успешно завершаются вовремя и в соответствии с их оценками. | Метрики производительности<br>Issue tracker<br>Система планирования<br>График расходования | Практики управления проектами (в том числе теория ограничений), практики agile-работы |

| | | |
|---|---|---|
| | буфера проекта | |

**Закончена**

Работа по производству результатов была закончена.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Все невыполненные задачи относятся к административным или подготовке к следующему куску работ. | Issue traker (все неадминистративные задачи закрыты) | |
| Результат работ был достигнут. | Воплощение системы готово | |
| Стейкхолдеры приняли результирующую инженерную систему. | Подписанный акт приёмки-сдачи | |

**Закрыта**

Все остающиеся служебные задачи были завершены, и работа была официально закрыта. Полученные уроки были сформулированы, записаны и обсуждены.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Метрики были сделаны доступными. | Адрес репозитория с журналами метрик производительности | Практика измерений |
| Всё было архивировано. | Адрес архивного репозитория с проектными документами | Практики архивирования |
| Бюджет был сверен и закрыт. | Акт о сверке и закрытии бюджета | Практики бюджетирования |
| Команда была освобождена. | Приказ о завершении работ | |
| Нет незавершённых, недоделанных задач. | Issue tracker пуст.<br>План проекта выполнен. | |

Технологии

Управление технологиями — это часть инженерного менеджмента, но иногда само название “инженерный менеджмент” приводят в варианте “инженерный менеджмент и управление технологиями”. Точнее было бы “управление практиками”, ибо смена ведущих дисциплин сюда тоже входит (практика = дисциплина + технология, технология = рабочие продукты + инструменты). Помним также, что “практику” самого верхнего уровня, достаточную для выполнения всех работ, называют методом разработки, или методологией разработки. Но уж не будем менять устоявшийся термин инженерного менеджмента на устоявшиеся термины ситуационной инженерии методов, просто будем помнить про разные варианты использования слова “технология”.

Огромное число инженерных технологий в современном производстве зафиксировано в виде широко распространённых стандартов и их групп (например, стандартов ЕСКД или СПДС) и поддерживающих их учебников и тренингов, регламентов (стандартов организации, описывающих порядок работы), программных продуктов. Но если коснуться клиентских альф и альф предпринятия, то определённость технологий улетучивается (кроме, пожалуй, технологий управления работами). Более того, определённость не так велика и в инженерии: если проект не требует использования ЕСКД или СПДС, или речь идёт о программистской части проекта, то становится трудно определить даже сами используемые практики: люди не склонны задумываться о практиках, в соответствии с которыми они работают.

Основные дисциплины в управлении технологиями: ситуационная инженерия методов (даёт понимание самого предмета технологии как объекта управления), архитектура предприятия (занимается описанием инструментария, поддерживающего практики работы) и организационные изменения (новые технологии требуют изменений в организации труда, в том числе и перераспределения полномочий), управление персоналом (образовательные программы по дисциплинам и организация тренинга по инструментам и рабочим продуктам).

Очевидная подальфа "технологии" — это "практика", а у практики могут быть в свою очередь выделены подальфы рабочего продукта и инструмента.

**Принципы установлены**

Принципы и ограничения, которые определяют технологию, установлены.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Команда придерживается принципов и ограничений. | Дисциплинированная команда | Образовательные практики для членов команды, найм квалифицированного персонала |
| Принципы и ограничения согласованы стейкхолдерами. | Утверждённый перечень дисциплин | |
| Потребные для работы инструменты и их стейкхолдеры согласованы. | Перечень инструментов, поддерживающих дисциплины | Архитектура предприятия |
| Рекомендации по избранному подходу доступны. | Литература в интранет<br>Стандарты<br>Нанятые консультанты | |
| Контекст, в котором будет работать команда, понятен. | Список отраслевых (например, ЕСКД или СПДС, или иностранные их варианты) и организационных стандартов | |
| Ограничения, которые применимы к выбору, | Планы организационного и технологического развития | Стратегирование (предпринимательские |

| | | |
|---|---|---|
| приобретению и использованию практик и инструментов, известны. | (организационная и техническая политика)<br>Бюджет<br>Критерии выбора инструментов | решения, работа с возможностями) |

**Основа положена**

Ключевые практики и инструменты, которые формируют основу технологии, выбраны и готовы к использованию.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Ключевые практики и инструменты, которые формируют основу технологии, выбраны. | Перечень практик (дисциплин+рабочих продуктов и поддерживающих их инструментов) | |
| Командой согласовано достаточное число практик, чтобы начать работу. | Перечень практик, результаты опроса | |
| Все не подлежащие обсуждению практики и инструменты были определены. | Перечень практик | |
| Неувязки, которые есть между доступными практиками и инструментами, и необходимыми практиками и инструментами, проанализированы и поняты. | План организационного и технологического развития | Архитектура предприятия с базисами as is и to be |
| Неувязки между тем, что нужно для выполнения желаемой технологии и уровнем возможностей команды, проанализированы и поняты. | План организационного и технологического развития<br>Описание текущих компетенций команды | |
| Выбранные практики и инструменты были интегрированы, чтобы сформировать технологию, которую можно использовать. | Вид жизненного цикла определён (практики систематически описаны)<br>Решения о вводе в эксплуатацию инструментов<br>Решения по утверждению форм рабочих продуктов | Ситуационная инженерия методов (карточки Essence), моделирование данных (форматы рабочих продуктов), инженерия предприятия (инструменты и поддержка практик) |

**Используется**

Некоторые члены команды используют технологию и адаптируют её.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Практики и инструменты используются для реальной работы. | Результаты опроса | Agile-couching (ответственные за работу с технологиями — “наставники”, “тренеры”, “советники”) |
| Использование выбранных практик и инструментов регулярно проверяется. | Результаты инспекций | Agile-couching (ответственные за работу с технологиями — “наставники”, “тренеры”, “советники”) |
| Практики и инструменты адаптированы к контексту команды. | Результаты экспертной оценки | Agile-couching (ответственные за работу с технологиями — “наставники”, “тренеры”, “советники”) |
| Использование практик и инструментов командой поддерживается. | Результаты опроса | |
| Процедуры для учёта обратной связи по поводу технологии работы команды – наличествуют. | Issue tracker<br>Регламент управления практиками<br>Результаты технологического мониторинга | Практики управления идеями, кружков качества, ретроспектив |
| Практики и инструменты поддерживают общение команды и сотрудничество. | График совещаний (в том числе дистантных)<br>Issue tracker<br>Возможности комментирования и другие возможности коллаборативной разработки в инструментах | |

**Наличествует**

Все члены команды используют технологию, чтобы выполнять свою работу.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Практики и инструменты используются всей командой для выполнения работы. | Результаты опроса | |
| Все члены команды имеют доступ к практиками и инструментам, требующимся для их работы. | Результаты опроса, результаты инспекций | |

| | | |
|---|---|---|
| Вся команда вовлечена в проверку и адаптацию технологии. | Результаты опроса | Практики управления идеями, кружков качества, ретроспектив |

**Работает хорошо**

Технология для команды работает хорошо.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Члены команды продвигаются в работе согласно плану, используя технологию и адаптируя её к их текущему контексту. | Результаты опроса | |
| Команда естественно применяет практики, не задумываясь о них. Инструменты естественно поддерживают способ работы команды. | Результаты опроса | |
| Команда постоянно подстраивает своё использование практик и инструментов. | Планы технологического развития | Практики управления идеями, кружков качества, ретроспектив |

**Вышла из употребления**

Технология больше не используется командой.

| Контрольные вопросы | Пример рабочих продуктов (views) | Пример практик описания (viewpoints) |
|---|---|---|
| Технология команды больше не используется. | Замещающая технология | Управление технологиями |
| Уроки опубликованы для использования в будущем. | Адрес репозитория с результатами ретроспективы | Управление технологиями |

*Пример введения новой альфы: подальфа «подрядчик»*

Если основные альфы в большом проекте недостаточны для нужной степени контроля, то нужно вводить дополнительные подальфы, и отслеживать их состояние. Для того, чтобы ввести дополнительную подальфу нужно:

- Убедиться, что это такой объект в проекте, за изменениями которого нужно следить.
- Назвать её и указать подчинённость (другим подальфам или основным альфам)
- Прописать список состояний альфы
- Прописать контрольные вопросы (checkpoints) для каждого состояния

Приведём пример подальфы "подрядчик" (вариант поставки оборудования для

непрерывного производства/process plant), это подальфа и "стейкхолдеров" и "команды" — ибо хотя подрядчик и внешний, но "играет за нас". За изменениями состояния подрядчика в проекте нужно следить очень внимательно, ибо забывчивость тут может очень дорого стоить. Итак, подрядчик:

*Признан*

- обоснование аутсорсинга есть
- требования и ограничения для подсистемы есть
- требования к гарантийному обслуживанию и ремонтным работам есть
- требования к технологиям работы есть
- сроки поставки определены
- критерии приёмки результата есть
- оценка возможной цены есть

*Выбран*

- технико-коммерческие предложения получены
- переговоры с кандидатами проведены
- технико-коммерческие предложения оценены
- подрядчик нами выбран

*Привлечён*

- валютные, сроков поставки, качества и прочие риски оценены
- договор с учтёнными рисками подписан
- аванс перечислен
- необходимая документация для начала работ передана
- уведомление о начале работ получено
- процедура коммуникации команды с исполнителями установлена

*Работы идут*

- надзор за работами установлен
- запросы обоих сторон учитываются
- запросы обоих сторон своевременно решаются
- график работ соблюдается
- технология работ соответствует ожидаемой
- коммуникация команды с исполнителем не прерывается

*Работы приняты*

- уведомление об окончании работ получено

- испытания проведены
- претензий к качеству работ нет (претензии сформулированы и устранены)
- акты проведения успешных испытаний подписан
- результаты работ доставлены на место
- документация передана
- монтажные работы произведены
- пуско-наладочные работы проведены

*Работы закрыты*

- акт приёмки-сдачи подписан
- деньги перечислены
- никаких дел к подрядчику нет
- гарантийные и сервисные работы производятся
- документация архивирована

## 10. Инженерия предпринятия

*Инженерия: организационная, предприятия, бизнеса, предпринятия*

Есть множество разных дисциплин, предметом которых является инженерия систем-организаций/предприятий/бизнесов. Так, organizational engineering — это академическая (научная) дисциплина, в которой речь идёт главным образом о "человеческой" (организационной) части предприятия — организации. Этот предмет организации очень грубо можно свести к способу распределения полномочий в альфе "команда": как команда делит между собой работы и координирует выполнение этих работ.

Enterprise engineering это практическая инженерная дисциплина, всё организационное дополняется в ней вниманием к технологической инфраструктуре, включая IT-инфраструктуру — предмет тут альфы команды, технологии, работы.

Много более широкий термин – это business engineering, поскольку в него включается также и стратегирование по выходу на какие-то рынки, т.е. предпринимательская деятельность: для бизнеса характерно внимание не только к организации и технологиям работы, но и альфе возможностей, которые приносят стейкхолдеры.

Системная инженерия в этом ряду дисциплин занимается своим делом: альфами определения и воплощения системы — замысливает целевую систему, проектирует её, изготавливает, испытывает, эксплуатирует и выводит из эксплуатации (хотя и необязательно все это происходит в рамках одного предпринятия).

Для того, чтобы системная инженерия могла заниматься целевой системой, нужно замыслить, спроектировать, изготовить, испытать, эксплуатировать, а затем модернизировать или вывести из эксплуатации предпринятие, как обеспечивающую систему: команду, технологии и выполняемые ими работы. Конечно, потребности в тех или иных компетенциях команды или потребности в тех или иных технологиях,

или потребности в выполнении тех или иных работ приходят из системной инженерии. Но вот удовлетворением этих потребностей занимаются инженерные менеджеры.

Сконцентрируемся на инженерии предпринятия — enterprise engineering, практической дисциплине. Она трактуется как поддисциплина системной инженерии, в которой к рассмотрению организации добавляется рассмотрение технологической инфраструктуры (включая IT-инфраструктуру, которая для современного предпринятия не менее важна, чем инфраструктура паровых машин в эпоху промышленной революции).

Тем самым тем самым инженерия предпринятия по своему предмету шире организационной инженерии (занимается технологиями), уже (на работу стратегирования) инженерии бизнеса, отличается по виду целевой системы от системной инженерии и не включает в себя лидерства (leadership) и стратегирования/маркетинга/продаж как традиционных менеджерских дисциплин.

Мы склоняемся к использованию термина «предпринятие» (от «что-то предпринять» — смесь значений enterprise и endeavour) как означающего организованных на достижение какой-то цели людей. Организованные — это с понятными полномочиями и ответственностями по имеющимся в их распоряжении трудом (возможность давать поручения друг другу), программами, оборудованием и объектами работы (рабочими продуктами).

Слово «предпринятие» позволяет унифицированно говорить не только о

- Предприятиях-юрлицах (включая холдинги или филиалы), но и
- об организованностях меньшего размера (подразделениях предприятий, рабочих группах, проектных командах — понятие классического “проекта” из проектного управления полностью укладывается в такое определение, равно как и в разы менее классические медицинские, менеджерские и инженерные “кейсы”), а также
- бОльшего размера (например, «расширенных предприятиях»/extended enterprise — работы нескольких сотрудничающих между собой предприятий в рамках одного большого проекта).

Тем самым *инженерия предпринятия* может пониматься как более широкое понятие, чем *инженерия предприятия* (т.е. инженерия организации одного юридического лица).

### Сообщества и их отличия от предпринятия: целенаправленная коллективная деятельность

Нельзя путать предпринятие (endeavour, effort, enterprise) с сообществами (communities), в том числе профессиональными сообществами (communities of practice — использующими какую-то практику).

Предпринятие преследует какую-то определённую цель и подразумевает кооперацию деятельности сотрудников предпринятия по созданию их целевых систем. Сообщества предполагают только обмен знаниями, выработку каких-то общих принципов, знакомство людей друг с другом, но они не создают никаких целевых систем. Есть целевая система (или последовательность создаваемых целевых систем) — есть предпринятие, нет целевой системы — нет предпринятия.

Ещё одно отличие предпринятия от сообщества — это организованность входящих

в него людей, понимаемая как возможность давать друг другу поручения по использованию команды и технологии предпринятия. Но, по большому счёту, это следствие того, что предпринятие имеет какую-то цель, поэтому люди должны как-то разделить труд по достижению этой цели и использовать инструменты и рабочие продукты. Сообществу это всё не нужно (хотя каждый из членов сообщества обычно входит в какое-то предпринятие).

О предпринятии не только можно, но и нужно думать как о системе, которая выполняет какую-то функцию: функцию, задаваемую собственниками предприятия (собственник — это стейкхолдер. Конечно, один и тот же исполнитель роли стейкхолдера может исполнять и роль собственника, и роль сотрудника, и даже роль клиента).

Типовая цель предпринятия — "деньги сейчас и в будущем", если речь идёт традиционном предприятии (если обобщить на предпринятие, то сразу всё не так просто — даже если в предпринятие входят кооперирующиеся в одном проекте две дочки холдинга, то "деньги сейчас и в будущем" должны бы относиться к холдингу, а не к каждой из дочек!).

А ещё предприятие должно обеспечивать прибыль владельцам, конкурентную зарплату сотрудникам, продавать целевые системы/сервисы клиентам. С предпринятием, которое иногда меньше, а иногда и существенно больше предприятия, всё ещё сложнее и запутаннее: любое предпринятие представляет собой систему систем, а поскольку люди ещё и "самопринадлежны" (собственники самих себя), то нельзя обычно однозначно указать на какую-то одну функцию, цель и миссию: разные люди по-разному её формулируют, в зависимости от своих убеждений, понимания, личных и позиционных ("застревания в стейкхолдерской роли") целей.

Предпринятие — это крайне сложная система систем, входящие в предпринятие люди обычно участвуют во многих предпринятиях, имеют в них разные полномочия, плюс сами предпринятия по-разному объединяются и декомпозируются в другие предпринятия. Обычно это не обсуждается в дисциплине "инженерия предприятия", но в силу увеличения глубины разделения труда уже невозможно разбираться только с тем, что происходит в рамках отдельного предприятия-юр.лица (с единством собственности, развития и операционного управления). Поэтому мы предпочитаем тут говорить об инженерии предпринятия: многие положения классической инженерии предприятия применимы и к предпринятиям самой разной природы (от рабочих групп по какой-то тематике внутри одного предприятия до межотраслевых проектов типа создания атомных станций), но для многих проблем приходится искать новые решения (когда речь идёт о координации деятельности в длинных межотраслевых цепочках поставок — например, при создании атомных станций).

Если даже от личных целей перейти к целям стейкхолдеров, то разные стейкхолдеры задают разные функции, цели, миссии предпринятия, нет однозначно одной цели и миссии — но чем более согласованы цели и миссии предпринятия между стейкхолдерами, тем проще дальше принимать решения по технологическому и организационному развитию предпринятия.

Также нужно помнить про два вида целей (это верно и при проектировании "железных", "программных" и разных других систем):

- Goals, которые никогда не достигнуть (деньги акционерам сейчас и в будущем — и побольше, побольше!) и про

- Targets, вполне достижимые (так, цель заплатить налоги есть, но вовсе не нужно их переплачивать. Цель заплатить зарплату сотрудникам есть, но не больше, чем нужно для работы предпринятия — если, конечно, сотрудники сами не акционеры).

Ещё нужно обратить внимание на цель самой инженерии предпринятия: её самая приоритетная цель получить быстрое во всех смыслах слова предпринятие — быстро выпускающее целевую систему, быстро перестраивающееся и осваивающее новые технологии, быстро выполняющее проекты. В конечном итоге в конкуренции побеждает обычно самый быстрый, поэтому цель повышения скорости работы явно выделяется изо всех других целей. Если вы не знаете, делать ли вам дешевле, лучше или быстрее — делайте быстрее, это в долгосрочном плане самая выигрышная стратегия (на сэкономленное время вы успеете сделать другие проекты, причём и лучше, и дешевле).

### Миссия предпринятия

Но вот миссия предпринятия формулируется в терминах его целевой деятельности, а не прибыльности. Знающие Элона Маска люди говорят, что заложить оранжерею на Марсе — это действительно определяемая им миссия SpaceX, а не просто миф для "поддержания бренда" и его пиарные рассказки для прессы (вся эта история со SpaceX началась с того, что разбогатевший во времена пузыря доткомов Элон Маск захотел купить ракету для доставки оранжереи на Марс. Но денег у него хватало только на одну ракету, а для доставки оранжереи нужно было минимум две. Он заинтересовался вопросом: почему ракеты такие дорогие? Так появилась компания SpaceX и текущая цель снизить стоимость запуска ракеты минимум вдесятеро). Церковь имеет формальную миссию "спасение", хотя не на секунду не забывает о деньгах.

Легко понять, говорите ли вы какие-то общие слова "из учебника" про цели и миссии, или о чём-то уникальном: попробуйте применить ваши формулировки к таким предпринятиям, как обувная фабрика или конструкторское бюро. Если это легко (для типичных "качественная продукция для широких масс" или "побольше денег, и прямо сейчас, с должным вниманием к экологии"), то вам нужно поработать над тем, чтобы сказать что-то специфичное именно для данного предпринятия, а не перепевать общеизвестные истины.

Инженерия предпринятия получает миссию предпринятия как входную для своей работы (это user needs, желания собственников).

### Корпоративное управление

Корпоративное управление (corporate governance, при этом слово governance лучше переводить не традиционно "управление", а "подотчётность", "обеспечение подконтрольности") — это практика признание ценностей и целей хозяев предпринятия, учет ограничений со стороны других стейкхолдеров, выбор текущих целей, выработка стратегии и тактики их достижения в изменяющихся внешних условиях (в том числе бизнес-планирование и составление программ развития — но таким образом, чтобы обеспечить подотчётность и подконтрольность предпринятия его хозяевам).

Мы не будем тут подробно останавливаться на этой дисциплине, она глубоко выходит за пределы инженерии предпринятия. Но инженер предпринятия должен строить предпринятие так, чтобы обеспечивать примат интересов хозяев над всеми другими интересами (интересы всех других должны быть удовлетворены на каком-

то уровне и дальше не нужно стараться в "перевыполнении плана", а вот интересы хозяев должны удовлетворяться настолько, насколько это возможно — нет ситуации "им уже хватит").

### *Стратегирование, маркетинг, продажи*

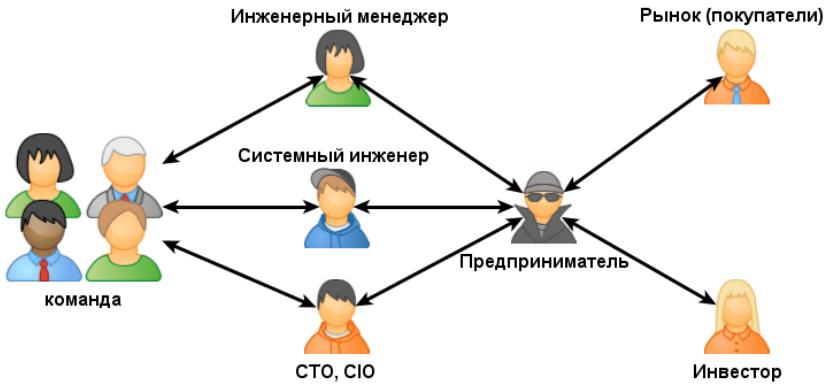
Чем будет заниматься предпринятие определяется в ходе практики стратегирования — многоуровневой деятельности с альфой стратегии (подальфе возможностей) и далее с определением предпринятия в части целеполагания (motivational model). Без стратегии предпринятию будет очень нервно: "либо вы планируете, либо вас планируют". Основной признак наличия стратегии — это прохождение теста Портера на сфокусированность (т.е. стратегия — это не "чем бы ещё заняться", а наоборот: нахождение такого занятия, при котором отказываешься от многих и многих других занятий).

Стратегией может быть всё что угодно (Генри Минцберг говорит, что стратегией называют любые из 5Ps — perspective, plan, ploy, pattern, position. Это и желаемое состояние мира в конце задуманного действия, и план действий, и хитрый трюк-уловка, и повторяющийся паттерн успешного действия, и выбор целевого рынка). Неважно даже использование слова "стратегия" в явном виде: неважно топ-менеджеры или собственники работают с альфой "возможности" и занимаются целеполаганием или все сотрудники по чуть-чуть (каждый на своём уровне удерживания целого, каждый в рамках своей специализации — но не теряя из виду системную целокупность предпринятия, насколько это возможно).

Стратегирование — это, безусловно, предпринимательская деятельность, умение "предвидеть будущее". Стратегирование (выбор того, чем заняться) по мере принятия различных решений (реализации стратегии) плавно перетекает в маркетинг (привлечение внешних стейкхолдеров-покупателей) и продажи (заключение конкретных сделок).

Нужно твёрдо запомнить: без этой деятельности инженерам нечего производить, им не будут платить зарплату, не будет понятно, какие технологии могут пригодиться в работе. Сначала стратегирование, потом инженерная деятельность. Конечно, инженеры могут подсказать стратегию, но если стратегия не сработала и маркетинг и продажи не принесли заказ на инженерный проект, то инженеру делать нечего, только просиживать штаны или юбку. Бизнес-архитектура решает всё.

По предпринимательству и стратегированию учебников нет, кто слишком много знает (а не делает) — тот обычно становится в этой сфере аналитиком, а стратегия требует таланта прежде всего в синтезе. Позиция предпринимателя крайне сложна, ибо нужно учесть интересы многих и многих стейкхолдеров, причём эти интересы должны быть синхронизированы во времени («вчера ещё рано, завтра уже поздно»):




Маркетинг тоже требует ярких идей, но уже более технологизируем.

Наиболее полезная книга для клиентской работы в продажах для инженерной работы — Нил Рэкхем “Стратегия работы с клиентами в больших продажах” (http://qame.ru/book/sale/strategiya_raboti_s_klientami_v_bolshih_prodazhah/%d0%a1%d1%82%d1%80%d0%b0%d1%82%d0%b5%d0%b3%d0%b8%d1%8f%20%d1%80%d0%b0%d0%b1%d0%be%d1%82%d1%8b%20%d1%81%20%d0%ba%d0%bb%d0%b8%d0%b5%d0%bd%d1%82%d0%b0%d0%bc%d0%b8%20%d0%b2%20%d0%b1%d0%be%d0%bb%d1%8c%d1%88%d0%b8%d1%85%20%d0%bf%d1%80%d0%be%d0%b4%d0%b0%d0%b6%d0%b0%d1%85.djvu), читать её нужно программой-читалкой файлов DjVu (например, WinDjView — http://www.koob.ru/about/windjview-setup.exe).

Инженерия предпринятия (как и системная инженерия) не занимается предпринимательством, стратегированием, маркетингом, продажами, но тесно связана с ними (так же, как и системная инженерия). Инженерия предпринятия создаёт ту часть предпринятия (организационную структуру и технологию, т.е. организационные звенья, инструменты, способы выполнения и координации работ, рабочие продкуты), которая занимается стратегированием/маркетингом/продажами — и обеспечивает неразрывную связь этой части предпринятия с инженерной частью.

### *Предпринятие как система-машина, а не толпа людей*

Чтобы получить в результате коллективной деятельности какую-то сложно организованную целевую систему (атомную станцию, медицинский лазер, коробок спичек), само предпринятие должно быть устроено как в каком-то смысле “машинная” система — эдакий сборочный конвейер, где на входе информация и сырьё, а на выходе цепочки определённых содержательных операций по обработке информации и сырья получаются готовые системы (продукты/товары, или сервисы, поскольку целевой системой вполне может быть инфраструктура самого предпринятия, продающего сервис).

Несмотря на то, что человеческие аспекты предпринятия тесно пересекаются с “машинными”, можно условно выделить этот “машинный” аспект предпринятия, спроектировать и “изготовить” его (т.е. закупить оборудование, нанять и обучить людей, организовать их для работы) как целевую систему. Именно этим “машинным” аспектом и занимается инженерия предпринятия — в отличие от других дисциплин (лидерства, классического менеджмента), в котором внимание акцентируется не на “машинных” аспектах, а как раз на человеческих аспектах.

В этом плане инженерия предпринятия вполне подобна системной инженерии — разве что:

- вместо инженерии требований по отношению к предпринятию говорят чаще о стратегировании/целеполагании (как и в случае инженерии требований, работа с user needs тесно связана с работой над требованиями — в данном случае определением функций предпринятия как "чёрного ящика", с выходом на функциональную декомпозицию), системную архитектуру называют архитектурой предприятия,
- по факту не говорят о проверках и приёмках, но занимаются измерениями (measurements — выставляют KPIs/Key Performance Indicators и проверяют их выполнение),
- задачи операционного управления (operation management — максимизации прохода работ через систему-предпринятие), лидерства (leadership) и управления организационными изменениями (change management) системы систем предприятия рассматривают совместно больше с менеджерской (т.е. учитывая человеческий и социальный аспекты предпринятия), нежели инженерной (предпринятие-машина) точки зрения.
- Классические представления системной инженерии не слишком применимы к предпринятию: в силу самопринадлежности каждого человека, как основной организационной единицы, речь всегда идёт о системо-системной инженерии. Это означает, что можно сколь угодно тщательно определять предпринятие, но вот воплощение предпринятия может проходить с огромным трудом: совместные организационные изменения частей предпринятия-с-людьми могут проходить не так легко и просто, как совместные изменения чисто технических систем, в состав которых не входят люди. Железка не разочаруется в жизни и не уйдёт в монастырь или к системе-конкуренту, а вот сотрудник, которого планировалось с его уникальными компетенциями использовать в каком-то определённом месте организационной "машины" — вот сотрудник это вполне может сделать.

Несмотря на все оговорки про непохожесть человеческого коллектива в его смеси со зданиями, оборудованием, программами на "машину", если инженер не определит (спроектирует) и затем надлежащим образом не воплотит такую систему-предпринятие, то вряд ли эта недоопределённая система-предпринятие сможет надлежащим образом выполнить свою функцию: толпа дикарей на острове не составит конструкторского бюро, разумное и глубокое разделение труда в этой толпе вряд ли проявится. И даже и толпа инженеров в центре мегаполиса тоже конструкторского бюро не составит: сначала нужно определить технологии проектирования, затем нужные для этой технологии квалификации людей, затем подготовить для них рабочие места, затем дообучить людей и/или нанять новых квалифицированных, затем организовать их (т.е. распределить полномочия согласно задуманному для реализации технологии разделению труда), и только потом можно ожидать выпуска продукта или оказания сервиса.

### *Развитие и совершенствование предпринятия*

Управление технологиями — это замена практик: осознание и вывод из использования старых практик и освоение новых практик (слово "внедрение" тут лучше не использовать: оно подразумевает активную позицию кого-то внешнего по отношению к осваивающим новую практику сотрудникам предпринятия и

пассивную позицию самих сотрудников. А ведь "лошадь нельзя заставить напиться, можно только подвести её к воде". Так и тут: нельзя "внедрить" какую-то технологию, но можно создать условия для её освоения).

Управление технологиями настолько важно, что специальность "инженерный менеджмент" иногда называют "инженерный менеджмент и управление технологиями" (engineering and technology management, изредка engineering, technology and innovation management). Есть множество университетских курсов, и даже научные журналы по этой тематике (например, http://www.mines.edu/ETM_GS, http://www.journals.elsevier.com/journal-of-engineering-and-technology-management/).

Нужно различать развитие и совершенствование предпринятия. Развитие — это смена технологии, совершенствование — это сохранение технологии. Развитие — это управление технологиями (переход от одних практик к другим), совершенствование — это отлаживание технологии, её "настройка", операционное управление, "зрелость процессов".

Развитие — это рост спектра возможных адекватных ответов человека или предпринятия на самые разные ситуации. Развитие — это когда всё реже и реже выдаются неадекватные реакции в *незнакомых* ситуациях из-за незнания и неумения, невладения инструментами, незнакомства с рабочими продуктами. Совершенствование — это когда всё реже и реже проявляются неуместные в знакомой ситуации поведенческие стереотипы человека или предпринятия. Развитие — это шаги вперёд, в новые ситуации. Совершенствование — это занятие круговой обороны в знакомой ситуации, и выигрыш за счёт этого. Развитие и совершенствование также легко спутать: например, при совершенствовании можно начинать замечать много нового в, казалось бы, знакомой ситуации — и это будет превращать совершенствование в развитие. Есть вариант и вообще не обращать внимания на разницу между развитием и совершенствованием, но я бы не рекомендовал это делать — как минимум, в учебном контексте.

Учебный же контекст есть всегда: и когда мы учим детишек (и сами устанавливаем баланс между их развитием и совершенствованием), и когда учим предпринятие (или оно само учится — learning organization Питера Сенджа, помним про "освоение" vs "внедрение"), и когда сами учимся жизни в ходе её проживания ("не перехождения поля").

Когда человек научается стоять, он не совершенствуется в этом бесконечно. Когда человек научается ходить, он не совершенствуется в этом бесконечно — но у него есть шанс стать победителем олимпийских игр по спортивной ходьбе. Когда человек научается бегать, то тоже можно или становиться чемпионом олимпийских игр по бегу. А можно учиться танцам, или вождению самолёта, или каким-то другим сложным двигательным практикам. Развитие это всегда лестница, на ней крутые ступеньки.

Проблема в развитии в том, что часто непонятно, зачем это — если нет зависти или любопытства к новым, более сложным навыкам. Бегунов и так неплохо кормят, как впрочем, и ходоков. Но оставим в стороне дискуссию о том, что заставляет развиваться человека. Предпринятия заставляет развиваться только конкуренция: если предпринятие не меняет свои технологии, то рано или поздно его продукция в силу совокупности причин (если нет "административного ресурса") перестанет покупаться — она будет или несовременна, или слишком дорога по сравнению с продуктами конкурентов.

Управление технологиями страшно, ибо прекращает совершенствование. Управление технологиями начинает с того, что показывает бесперспективность совершенствования в применяющихся на предпринятии практиках стратегирования/маркетинга, операционного управления, инженерных практиках. Контрольный вопрос для управления технологиями не в том, какие новые практики будут освоены, а какие старые практики будут прекращены.

Совершенствование комфортно и обманчиво безопасно. Если жизненные или рыночные ситуации не имеют шанса поменяться, то совершенствование — верный путь к успеху. Если жизненные ситуации меняются, то тратить время на совершенствование — смертельно. Нужно найти в себе силы перейти к развитию, оторвать задницу (свою или предпринятия) от приятного и общественно поощряемого.

Но без совершенствования нет ступенек на лестнице развития. Не закрепившись на какой-то ступеньке, невозможно шагнуть вперёд. Если нет опыта какой-то деятельности, то на эту деятельность нельзя будет опереться и шагнуть дальше. Поэтому нужно прекращать ежедневно прыгать с деятельности на деятельность в "ознакомительном" режиме. Если вы круто переразвиты, но при этом ни в чём не совершенны — не удивляйтесь, что у вас непрекращающиеся проблемы, вам одинаково будет тяжело и в знакомых ситуациях, и в незнакомых.

Так что же делать? Как найти правильное время пребывания на каждой образовательной/технологической ступеньке — правильное время совершенствования какой-то практики? Общих рецептов тут нет. Но над этим нужно специально думать, нужно кроме утопания в тактическом совершенствовательном планировании постоянно заниматься развивательным стратегированием (альфа технологий существенно связана с альфой “возможности”: возможность выполнить проект появляются не только тогда, когда клиенты *хотят*, но и когда предпринятие *может*!), и наоборот — не забывать кроме вечного маниловского стратегирования ещё и что-то планировать, чтобы быть совершенным.

### Проект технологического развития: постановка практик

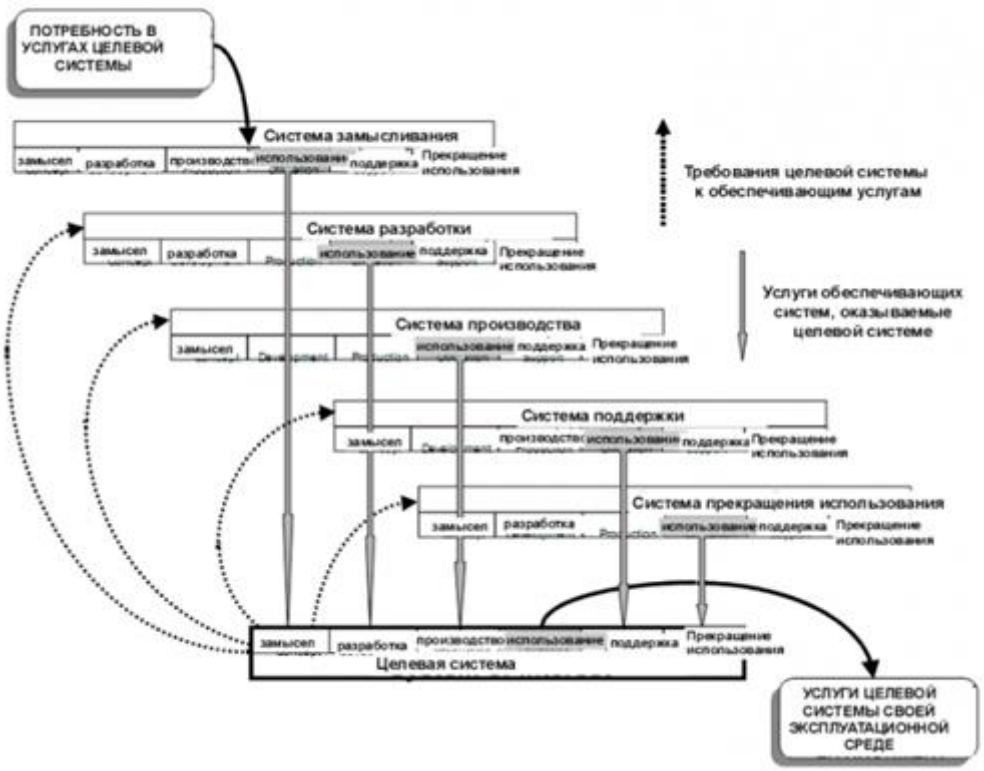
Помним театральную метафору: каждый и внутренний и внешний по отношению к команде проекта стейкхолдер играет свою роль в спектакле инженерного проекта, у него определённые работы по определённым технологиям в предпринятии. Но для того, чтобы актёры смогли совместно сыграть спекталь, его *ставят* — закупают реквизит и шьют новые костюмы, изготавливают декорации, актёры учат роли, потом долго репетируют. То же самое происходит в ходе управления технологиями: для новой технологии закупают инструменты, разрабатывают форматы новых продуктов, исполнители учат новую дисциплину и получают навыки работы с новыми инструментами и видами рабочих продкутов (а иногда приходится приглашать новых исполнителей: не всех извозчиков берут в таксисты).

У предприятия различают производственные (целевые) проекты и проекты развития. “Производственные” — это просто чтобы отличить проекты по оказанию услуг клиентам, проектированию, конструированию, изготовлению целевых для клиентов систем. В системно-мыследеятельностной методологии предпочитают говорить даже “воспроизводство” — чтобы показать неизменность, повторяемость, цикличную воспроизводимость практики: неизменность дисциплины, неизменность инструментов, неизменность видов рабочих продуктов. То есть используемое слово “производство” (producing, production) включает и замысел, и разработку, и изготовление, и испытания, и эксплуатацию, и вывод из эксплуатации целевой

системы — всё, кроме развития, понимаемого как развитие обеспечивающей системы, развития предпринятия, т.е. изменения технологий (и по сопричастности изменения полномочий — организационное развитие).

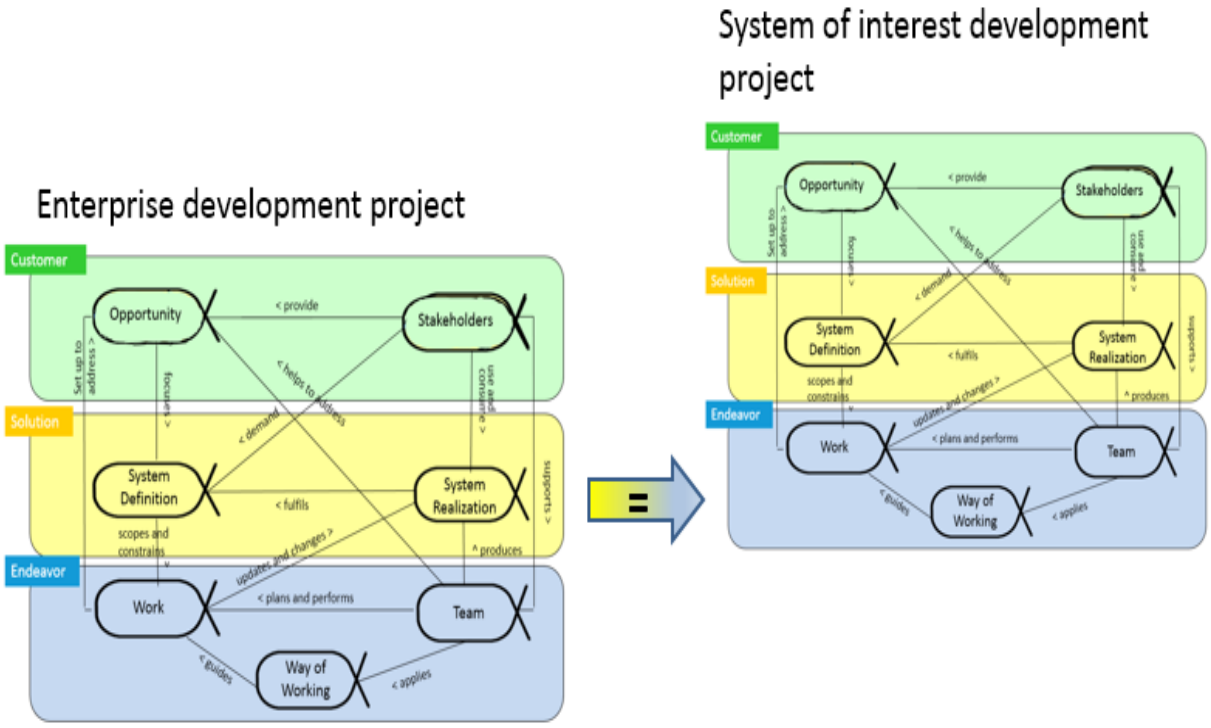
Целевой системой инженерного (инженерии предпринятия!) проекта развития является технология производственного проекта (в части разворачивания новых инструментов и рабочих продуктов) и команда производственного проекта (в части образования членов команды по новым дисциплинам и научения их работе с инструментами и рабочими продуктами). Предпринятие развития является обеспечивающей системой для производственного предпринятия.

В ISO 15288 это изображается так:




“Целевая система” это инженерное предпринятие в части производства, его услуги (оказываемое вовне поведение системы) операционному окружению — это выпуск продукции, а системы замысливания-разработки-поддержки и т.д. это системы управления технологиями (службы стратегирования, отделы развития, консультанты по отдельным дисциплинам и инструментам и т.д.).

Более удобно и просто показывать разделение проектов развития (организационных изменений, управления технологиями) и производственных проектов (создания целевых систем/продуктов/изделий/сервисов) на диаграмме альф инженерного проекта:




Инженерное решение проекта развития — это производственное предпринятие.

Не забываем, что стейкхолдер — это только роль. Исполнитель разных ролей может быть одним и тем же человеком (или предпринятием — проектной командой, рабочей группой, подразделением). Так что одни и те же люди могут и заниматься и производством, и совершенствованием, и развитием. Одной и той же ручкой, одним и тем же Word можно писать тексты и по проектам развития, и по производственным проектам. Но ввиду глубокого разделения труда вполне возможны ситуации, когда главными в развитии будут не-производственники, а специально выделенные люди: отдел развития, приглашённые консультанты. Конечно, производственники тоже участвуют при этом в работе, но они будут… рабочими продуктами. Помним о лидерстве (leadership): это дисциплина, которая позволяет людям добровольно становиться рабочими продуктами в проектах развития и не менее добровольно занимать позиции в производственных (целевых) проектах. Именно лидерство объединяет инженерию предпринятия как инженерную дисциплину и традиционный менеджмент как социологическую и психологическую дисциплину.

Организационное развитие. Закон Конвея

Закон Конвея (http://www.melconway.com/Home/Conways_Law.html) утверждает, что структуры определения целевой системы/продукта/изделия/сложного инженерного объекта/сервиса будут копировать структуры коммуникации создающего его предпринятия. Другими словами, два модуля не смогут провзаимодействовать, если не смогут провзаимодействовать их создатели — поэтому для заработавшей системы структура конечных модулей будет отражать структуру коммуникации разработчиков. Структура целевой и обеспечивающей системы оказываются связанными. Плохо, если разбиение системы на модули игнорирует коммуникационную и организационную структуру разработчиков этих модулей: если одно подразделение проектирует фюзеляж и полдвигателя самолёта, а второе подразделение вторые полдвигателя и кусок крыла, то так

спроектированный самолёт не взлетит.

Фред Брукс сформулировал следствие из этого закона: поскольку первые приходящие в голову архитектуры обычно плохи, то разбиение системы на модули неизбежно будет меняться по ходу дела. Это означает, что организация должна будет отражать изменение архитектуры, она должна быть гибкой — и чем более гибка организация в изменении своей структуры, тем более хороши будут архитектуры, тем проще будет выжить организации в ходе конкурентной борьбы.

Лидерами рынка обычно являются те организации, которые приходят на рынок с новыми архитектурами — это означает, что лидерами рынка будут организации, которые сумеют быстро менять свои структуры в соответствии с новыми архитектурными решениями. В соответствии с законом Конвея изменить архитектуру целевой системы — это означает и изменить архитектуру предпринятия, которое занимается этой системой.

Современные предпринятия меняются непрерывно. Вот визуализация подобной перманентой реорганизации для Autodesk, отражающая постоянные кадровые перестановки, позволяющие компании выдерживать конкурентоспособность продуктной линейки: http://www.youtube.com/watch?v=mkJ-Uy5dt5g

Видеоролик воспроизводит 1498 дней организационной иерархии компании Autodesk (с мая 2007 по июнь 2011), каждая секунда — это примерно одна неделя. Каждый сотрудник представлен кружком, бОльшие кружки обозначают сотрудников с бОльшим числом подчинённых. Каждая линия означает подчинённость одного сотрудника другому (кто чей начальник), дерево подчинения дальше анимировано, чтобы показать изменения (полная информация по проекту — http://www.autodeskresearch.com/projects/orgorgchart). Видео там высокого разрешения, смотрите на максимально доступном разрешении.

Не нужно думать, что такой темп кадровых перестановок некомфортен для работников, у которых постоянно меняется их подчинённость и вовлечённость в разные проекты. Autodesk занимает 54 место в списке Fortune 100 компаний-лучших работодателей, http://money.cnn.com/magazines/fortune/best-companies/2013/snapshots/54.html — несмотря на эти постоянные кадровые перестановки.

Конечно, чтобы поддерживать порядок при таком темпе изменений, необходимы компьютерные системы — именно они помнят, кто кому когда подчинялся, кому какую платить зарплату, кто когда уходил в отпуск, кто на какой проект работал. Лет двадцать назад про IBM писали, что одновременно в любой момент в ней идут в среднем 35 проектов реорганизации, и это число казалось запредельно большим, а опыт IBM уникальным. Сегодня это уже поддержанная компьютерами норма жизни, так живут уже все крупные компании.

Упражнение: для тех компаний, в которых вы работали, подумайте: хватит ли информации в компьютерной системе предприятия, чтобы сделать такую визуализацию хода перестановок, какую сделал Autodesk? Какой темп организационных изменений в этой организации? Какой темп смены продуктных линий, сопровождается ли он реорганизациями?

### Системноинженерное мышление и инженерия предпринятия

Все наработки системного подхода вполне приложимы к предпринятиям-системам:

- Предпринятия материальны (согласно многим стандартам

предпринятие/организация определяется как люди, здания, сооружения, оборудование и другие ресурсы, причём среди людей есть договорённости по полномочиям в использовании этих ресурсов).

- Предпринятия состоят из структурных иерархий компонент, модулей, размещений (хотя о них говорят в такой терминологии редко, но подобного типа мышление сохраняется). Архитектура предпринятия (для современного предпринятия при этом обязательно подробное рассмотрение информационных систем в составе архитектуры предприятия) уже давно является одной из признанных дисциплин. Но, конечно, "принципиальные схемы" у предпринятия отличаются от оптических и гидравлических схем.
- Предпринятие имеет жизненный цикл (но тут нужно обратить внимание на особенность: как барон Мюнхгаузен вытягивал себя за волосы, так и предпринятие обычно само себя протаскивает через жизненный цикл: оно само себе обеспечивающая система и целевая система: само себя проектирует/стратегирует, само себя строит, само себя эксплуатирует — но всё-таки различают производственную деятельность предпринятия и деятельность по развитию предпринятия).

Конечно, терминология отличается (хотя стейкхолдера назовут стейкхолдером и архитектуру архитектурой), но мы не зря тренировались абстрагироваться от конкретных слов и больше заниматься смыслом говоримого.

Тем самым весь материал предыдущих разделов вполне применим для обсуждения предпринятия.

Совершенствование предпринятия — это необходимость практиковать какую-то ступень зрелости практик, чтобы получить хоть какой-то шанс шагнуть вперёд, на следующую ступень развития. Совершенствование зрелой уже практики — это часто путь к загниванию, это прекращение развития. После того, как вы довели до блеска бумажный документооборот, у вас могут быть огромные проблемы с развитием в сторону электронной коллаборации, ибо там не просто "электронный документооборот", но проблемы с самим понятием документа как универсальной сущности — речь ведь идёт главным образом о совместной работе разных приложений с общими базами данных! Отказаться от документов в этот момент — это будет восприниматься не как развитие, а как махровое антисовершенствование, разрушение основ и крах того совершенства, которым нужно гордиться! Совершенное воспринимается, как лучшее, а результат развития — только как хорошее (ибо всем очевидно, что этому ещё совершенствоваться и совершенствоваться, чтобы сравниться с уже имеющимся). "Хорошее — враг лучшего!" — и развитие прекращается, не начавшись. Старые парадигмы, старые cognitive frameworks умирают только с их носителями...

Первый шаг решения проблемы "развитие против совершенствование" — это возможность её коллективного (ага, в том числе в коллективе с самим собой! Попробуйте написать ваши мысли, а затем прочесть — и при написании, и при прочтении вы откроете для себя много нового!) обсуждения.

Для самой возможности обсуждения совершенствования и развития нужно опереться на системноинженерное мышление. Harold Lawson писал в комьюнити INCOSE в LinkedIn в треде "Building SE into an Organization", — "...it is all about understanding and communication. If you do not get people on the same page you are simply talking in the air. That is why learning fundamental paradigms, concepts and principles of systems is so important". Развитие начинается в тот момент, когда его

можно обсуждать, когда выучен язык, на котором говорят о развитии. Нужно освоить "fundamental paradigms, concepts and principles of systems" — и пройти ступеньку совершенствования, чтобы научиться этим пользоваться. Нужно сделать шаг "развития для развития". Это трудно: прекратить индивидуальное или корпоративное совершенствование в том, чем вы прямо сейчас сами или с коллегами заняты, и пройти ряд образовательных ступенек, которые с точки зрения текущей ступеньки "ни о чём".

Но эти "ступеньки ни о чём" лежат в основе любого общего образования, которое готовит развитых людей — т.е. людей, не теряющихся в широком разнообразии ситуаций. Это совсем не "совершенствование", которое имеет своим предметом что-то предельно конкретное, и результаты которого довольно легко измерить (ибо совершенствование — это повышенная эффективность поведения в одной и той же ситуации).

Самое эффективное для развития предпринятия — это не сразу бросаться в преобразования и менять технологическое шило на технологическое мыло, а сначала научить людей говорить об их предпринятии, научить людей говорить о стратегировании/развитии на языке системноинженерного подхода. Люди получают возможность быстро договориться между собой и скооперироваться (в том числе договориться "внутри себя с собой", если ваше предпринятие — это только вы сами). И тогда начинают происходить чудеса, тогда начинается настоящее развитие и осмысленное для него совершенствование — в маркетинге, инженерии, операциях.

### Цикл непрерывного совершенствования

Совершенствование нельзя недооценивать. Так, использование цикла непрерывного совершенствования теории ограничений обычно за первые несколько месяцев даёт повышение производительности в текущей деятельности примерно на треть!

В книге Элияху Голдратта “Цель: процесс непрерывного совершенствования” и в других книгах этого автора (ознакомительные экземпляры вы можете найти тут: http://rutracker.org/forum/viewtopic.php?t=2403312 — не забудьте потом купить эти книги!) показывается, как работать с логистической метафорой “потока” в операционном управлении: предприятие представляется чем-то типа дельты реки, со множеством ручейков, запруд, каналов, перетоков — там, где стоит “запруда” (по разным причинам медленно пропускающее через себя рабочие продукты организационное звено),там образуется “очередь на обработку” — и необходимые для включения в состав готовой целевой системы рабочие продукты застревают в этой очереди, задерживая готовность целевой системы в целом.

Главная задача менеджера (операционного управляющего) — это непрерывно прочищать “запруды”, максимально влияющие на задержки по выпуску основной системы. Главная задача — это увеличение “прохода” (протока) через предпринятие, ибо если целевые системы (оказываемые сервисы) будут быстрее и в большем количестве уходить, то и деньги за них будут приходить быстрее и в большем количестве (про особенности управленческого финансового учёта в теории ограничений см. книгу Томаса Корбета “Учёт прохода”/”Throughput accounting”).

В ходе выполнения работ по множеству проектов предпринятия вы должны находиться в цикле непрерывных улучшений:

1. Найти ограничение (операцию, которая ввиду недостаточности ресурсов держит весь выпуск готовых систем/продуктов/услуг). Это не так просто, ибо ограничением может быть что угодно: станок малой производительности, чья-то сверхценная голова, запрещающие что-то необходимое для ускорения правила регламентов, излишняя тщательность ведения какой-то работы. Обычно то, что является ограничением подтверждается каким-то расчётом (это очень похоже на физические расчёты по протоку жидкости по системе трубопроводов разной пропускной способности, поэтому один из подходов к таким расчётам так и называется Factory physics — "физика фабрики").

2. Максимизировать проход через ограничение (снять ограничение оказывается обычно очень долго и дорого), т.е. эксплуатировать дефицитный ресурс на 100%, избегая его простоев и поломок, потери времени на брак и т.д.

3. Подчини всю остальную работу ограничению — так, чтобы "не перевыполнялись планы" (всё равно лишние результаты работ не пройдут через ограничение! Зачем же тогда тратить драгоценные ресурсы?). Обычно это самая трудная, контринтуитивная часть: тут нужно не столько "больше работать", сколько прекратить работать!

4. Вернуться к началу цикла (часто даже не нужно снимать ограничение: максимальная эксплуатация ограничения и подчинение всех остальных работ этому ограничению приводят к тому, что узкое место в потоке перемещается в другое место предпринятия).

### Цикл Деминга

Ещё один широко известный цикл совершенствования — Шухарта-Деминга "Plan, Do, Check, Act" (PDCA):

- планируй работу (Plan — найди проблему),
- работай (Do — устрани проблему),
- проверь (Check — проверь качество, измерь результат),
- действуй (Act — проведи коррекцию: меры по устранению выявленных при проверке проблем),
- действуй дальше с начала цикла.

Он уже не столько цикл управления операциями (альфа работы), сколько управления технологиями (в части совершенствования технологий), используемый для управления качеством как уменьшением разброса в характеристиках продуктов. У.Шухарт впервые описал концепцию PDCA в 1939 г. в своей книге "Статистические методы с точки зрения управления качеством". Э.Деминг пропагандировал использование цикла PDCA в качестве основного способа улучшения качества: постоянная переналадка процессов работы с тем, чтобы устранить "неслучайные/систематические отклонения".

Основная идея этого цикла в том, что разброс параметров выпускаемой продукции (несоответствие ожиданиям однородности выпуска продукции с заданными параметрами) обусловлен как проявлением каких-то неслучайных причин, так и проявлением чистых случайностей. Если заготовка для обработки поступает грязная, причина плохого качества будет совсем не случайной, эту причину можно найти и устранить (например, промыть заготовку перед обработкой, или упаковать для того, чтобы она не приходила грязной). А вот если плохое качество получилось

из-за того, что скакнуло напряжение в обычно стабильной сети питания станка — то это чистая случайность. Но если напряжение скачет постоянно, то и эту причину плохого качества можно устранить: поставить стабилизатор питания для станка, или заведомо выкидывать ту дешёвую заготовку, во время обработки которой произошёл сбой (а не пропускать её в выходную продукцию). Для того, чтобы поднять качество, нужно не столько просто его потребовать, объявив какие-то "жёсткие допуски", сколько непрерывно искать источники проблем и устранять их, оставляя место только случайностям.

#### Шесть Сигм

Ещё одна практика, которая занята именно совершенствованием в части управления качеством через управление технологией работы и использование статистики — это Шесть Сигм (six sigma, http://en.wikipedia.org/wiki/Six_Sigma).

Название означает ситуацию, при которой статистически (все технологии работы с качеством опираются на статистику!) на 1 миллион изделий приходится 3.6 дефекта.

Как и любое другое совершенствование, практика шести сигм заключается в непрерывном прокручивании цикла DMAIC (Define, Measure, Analyze, Improve, Control), а в части проектирования (DFSS, Design for Six Sigma) цикла DMDV (Define, Measure, Analyze, Design, Verify).

Необязательно тут даже разбираться с содержанием всех этих циклов "обеспечения качества" (они удивительно похожи на вариации цикла Дёминга) — но нужно запомнить, что любое совершенствование циклично, по единичке качества за цикл.

Современный вариант шести сигм называют Lean Six Sigma (где lean переводится очень по-разному, вплоть до "бережливости" и обычно означает минимизацию работы в духе agile — если что-то можно не делать, то в lean это не делается).

В практике шести сигм можно найти целый набор различных практик, который в их приложении к какой-то технологии даёт её улучшение.

Критика Голдратта к шести сигмам была очень простой: он говорил о необходимости сначала находить ограничение (т.е. подходить сначала не с позиции "управления качеством", а с логистической позиции "увеличения прохода работ через предпринятие"), и уже затем применять всю мощь практики шести сигма к найденному ограничению.

Тем самым нужно запомнить, что все эти "циклы совершенствования" обычно вложены друг в друга и взаимосвязаны (так, есть ещё цикл совершенствования процессов по "ступенькам зрелости", стратегический цикл Бойда и т.д.).

### *Архитектура предпринятия*

#### Основные альфы организационного и технологического решения предпринятия

Условно можно любое предпринятие вообразить выполняющим какие-то инженерные проекты. Предприятие что-то замышляет, проектирует, создаёт,

отлаживает, эксплуатирует (а не занимается фундаментальными исследованиями — хотя и в этом случае можно представить себе жизненный цикл теории, появляющейся из гипотезы и проверочных экспериментов, но это будет большой натяжкой. Системные инженеры есть, системных учёных нет). В принципе, даже спасение душ в ходе прохождения каких-то тщательно спроектированных ритуалов в церкви — это ведь тоже вариант инженерной деятельности, привнесение в реальный мир из мира идей вполне материальных людей с определённым той или иной идеологией спасения образом мыслей!

В принципе, все такие инженерные деятельности можно также переформулировать как "сервисы". Так, благотворительный фонд — это сервис для благотворителей, но вот бенефициары у него — получатели благ, они бенефициары, но не клиенты!

Если переформулировать систему-предпринятие в сервис, это оказывается чрезвычайно плодотворным. Предприятие, продуктом которого являются чугунные чушки мыслится совсем по-другому, чем предпринятие у которого сервис "поставка чугунных чушек". Ибо сами чушки много менее разнообразный объект, нежели сервис "поставка" (в этом сервисе можно варьировать сроки поставки, распределение поставки во времени — чтобы она коррелировала с распределением потребления чушек по времени, включать "дополнительное обслуживание" типа логистики, в том числе погрузочно-разгрузочных операций и т.д.).

Как и любая другая система, предпринятие имеет определение и воплощение. Мы не говорим обычно, что это альфы "инженерного решения предпринятия", это легко путается с инженерным решением целевой системы предпринятия. Скорее, мы говорим тут об альфах организационного и технологического решения предпринятия: как распределены ответственности и полномочия между людьми и группами людей (подразделениями) и какие практики поставлены (технологии развёрнуты) на предпринятии.

### Подальфы определения предпринятия

Определение предпринятия имеет подальфы:

- Стратегии, описания целеполагания, бизнес-архитектуры (примерный аналог "требований" — описание системы в терминах её поведения как чёрного ящика)
- Архитектуры предпринятия (основные организационные и логистические решения)
- Неархитектурных организационных и логистических (операционных) решений.

Как и в случае и просто инженерной системы, описание системы-предпринятия для этих определений может быть, но может и не быть (так, описание может прозвучать на каком-то из совещаний менеджмента, или часть предпринятия как-то "сложилась" и никто не потрудился её описать, или только что описанное предпринятие быстро поменялось — ведь в предпринятии есть живые люди, это не железная или программная система, которая не может поменяться самопроизвольно!).

На определение предпринятия есть две полярных точки зрения:

- бесполезно его выражать в рабочих продуктах, ибо оно никогда-никогда не будет совпадать с реальностью. Люди несамотождественны в разные моменты времени! Просто описание предпринятия меняет его (ибо люди,

получив эти описания, немедленно меняют свою деятельность с учётом этой новой информации).

- Мы ничего не сможем изменить-поменять в предпринятии, пока мы его хоть как-то не определим (может быть, не детально, но в основных чертах — основные цели/требования, архитектуру).

Как обычно, определение предпринятия выражается в виде групп тематических описаний, которые в свою очередь состоят из моделей. Так, архитекторы предприятий признают стандарт ISO 42010 как относящийся и к архитектуре предпринятия (не только к программной или системной архитектуре).

### Подальфы воплощения предпринятия

Определить/описать предпринятие — это описать его в терминах различных подальф его воплощения:

- деятельность (компоненты: услуги, сервисы, функции, роли, полномочия) — как устроено предпринятие, как оно работает (run time).
- Подразделения (как сгруппированы ресурсы — люди-сотрудники, здания, оборудование, материалы: модульное описание). Органиграмма и штатное описание, инвентарные ведомости для оборудования — это как раз модульное описание.
- Размещение (географическое нахождение ресурсов предпринятия): распределение подразделений (людей и оборудования) по зданиям, странам, рынкам и т.д.
- Многие другие интересные разным стейкхолдерам аспекты, ибо помним, что разные стейкхолдеры видят разные части в одном и том же объекте — части, удобные для использования в их деятельностях.

### Виды практик описания деятельности

Деятельность предпринятия (”принципиальная схема”, “как оно работает”, компонентное описание) может быть описана самыми разными практиками описаний (viewpoints), делающими акценты на разные альфы в зависимости от того, для какой “деятельности над деятельностью” эти описания будут полезны:

- Технологии (way of working) — для управления технологиями (постановки практик, определения нужных компетенций, понимания связи с инженерией — какие операции нужны, какие рабочие продукты, какие инструменты) нужны описания деятельности как состоящей из выполняемых практик — дисциплины (альфы), рабочие продукты и инструменты. Упор на то, с чем работаем (альфы и рабочие продукты), и только потом — какие с ними операции и в какой последовательности. Типичный пример — это описания ситуационной инженерии методов (в том числе по стандарту OMG Essence, но также и SPEM 2.0 и т.д.).
- Работы — упор на логистику: отслеживание выполнения работ ограниченными ресурсами предпринятия с целью максимизации прохода работ и контроля выполнения всех необходимых работ. Обычно это “процессные” и “проектные” описания.
- Команда — упор на организацию тех, кто выполняет работы, задаваемые практиками (полномочия, поручения, обещания и т.д.). Помним также, что

каждый член команды — это также и стейкхолдер, так что это описание "полномочия ролей" (компонент, а не заполняющих эти роли людей и подразделений).

Это, конечно, очень грубое деление — и нужно помнить, что и альфы тесно связаны друг с другом, и описания чаще всего гибридны. В работе http://www.ww.bptrends.com/publicationfiles/01-09-ART-%20Case%20Management-1-DeMan.%20doc—final.pdf эти основные виды моделирования называются соответственно:

- Artifact-based (упор на рабочие продукты — разные операции работают с какими-то рабочими продуктами),
- activity-based (процессно-ориентированные с предопределённой последовательностью операций),
- communication- (or conversation-) based: цель деятельности достигается как результат серии взаимодействий (коммуникаций, переговоров) участников деятельности.

Предпринятия-киборги, workflow

Когда-то считалось, что киборгами (cyborg = cybernetic organism) прежде всего станут люди: половина мозга и органов у них будут машинными, а половина останется "традиционными человечьими". Но случилось так, что киборгами стали сначала организации: половина работы в них выполняется людьми (в том числе работы со знаниями и информацией), а половина уже выполняется инструментами (в том числе информационными системами).

Компьютер помнит сейчас планы работы из тысяч позиций, позволяет налаживать учёт конфигурации систем с миллионами компонент и модулей, ведёт расчёты графиков работы, хранит накопленные знания по прошлым проектам — без компьютеров современные организации просто не смогли бы выполнять сложные работы с участием тысяч людей и миллионов комплектующих, причём делать это дёшево и быстро.

Часто про деятельность, выполняемую людьми с использованием информационных систем, говорят "ход работ" (workflow — обратите внимание, что у нас работы, деньги, время "идут", а в английском они "текут" — поэтому в русском языке труднее представить себе "потоковую" метафору предпринятия, необходимую для понимания операционного менеджента). В работе http://www.personal.psu.edu/axk41/BPM05-jerry-reprint.pdf для хода работ выделяют те же самые три "перспективы" (т.е. viewpoint), только называют их по-другому:

- для технологий информация-ориентированные (information-based, ибо речь идёт об информационных рабочих продуктах главным образом — отчётах, документах, формах и т.д.),
- для работ процесс-ориентированные (process based), а
- для команды — организационно-ориентированные (organization-based).

Как минимум, нужно помнить, что "процессный подход" (и его варианты "управление проектами", "управление делами/case management") абсолютно недостаточны для описания деятельности, для описания хода работ, для описания предпринятия в целом — есть и много других подходов, ставящих акценты и на альфы с рабочими продуктами (когда описывается не логистика, а содержание

деятельности) и на координационные акты в команде (организационно-ориентированные).

Организация, координация, коммуникация

Не любую группу людей, зданий, оборудования, материалов называют организацией — но только лишь ту, в которой оговорены полномочия людей по распоряжению этими ресурсами, включая возможности поручать друг другу выполнение работ. Это и есть сущность организации — коммуникация (разговор, общение) в части координации выполнения работ: обсуждение поручений, обещаний, согласий. Все эти "обещания" и "поручения" — речевые действия/speech acts: то, что происходит в речи, но влияет на ситуацию в реальном мире. Язык, оказывается, может не только описывать мир, но и как-то изменять его — http://plato.stanford.edu/entries/speech-acts/).

Наиболее развиты положения теории речевых действий в отношении к инженерии предприятий оказались в подходе DEMO ( http://www.demo.nl/, на русском языке см. краткое изложение метода в http://www.techinvestlab.ru/files/504456/demo_praxos_1.doc). В DEMO все действия различаются на производственные (действия по изменению вещества или информации) и координационные (поручения работы, обещания выполнить работу, сообщение о выполнении работы, подтверждение выполнения работы, отказ от выполнения или приёмки работы).

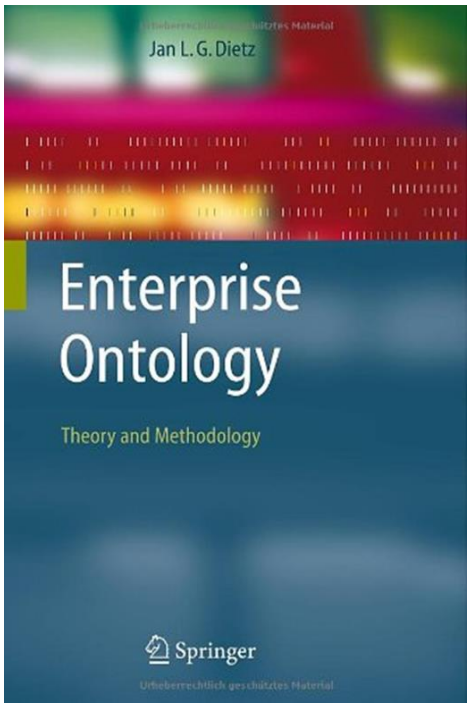
Исполнители работ (в демо называемые actor, актёр, исполнитель — помним театральную метафору!) имеют как компетенции по выполнению производственных действий (или отказу от них), так и полномочия по выполнению координационных действий. Каждое действие в DEMO называется трансакцией и проходит по следующему циклу, даже если некоторые элементы этого цикла и заданы неявно (хотя рекомендовано задавать их явно: неявное задание часто приводит к ошибкам в понимании ситуации): инициатор трансакции желает получить какие-то результаты, затем просит их у выполнителя работ, выполнитель работ обещает выдать результаты, затем производит их, затем объявляет о готовности результатов работы, затем инициатор работ принимает результат работы, затем инициатор использует результаты работы в своей части производства.

В какой-то мере "организовывание" — это прорабатывание вопроса о координации работ, проектирование и последующее воплощение схемы распределения полномочий в части координационных действий (кто кому может поручать какие работы, так, чтобы вероятность отказа была мала и любые отказы были обоснованно аргументированы). Тем самым исполнитель кроме функциональной его роли стейкхолдера (архитектор, инженер, слесарь) в организации имеет ещё и координационную роль (какого сорта работы он обычно поручает и принимает их результаты, а какие берётся исполнять и сдавать их результаты).

В значительной мере "документооборот", "процессы", визирования, управления поручениями — это не только и не столько содержательное выполнение самого дела (инженерная/техническая деятельность как таковая — изменение информации и вещества по мере их превращения в целевую систему), сколько координационная деятельность в рамках "организации": выражение просьб, обещаний, предъявлений выполненной работы, акцепта результатов, отказов от выполнения или акцепта.

Jan Dietz утверждает, что это и есть то, из чего по своей сути состоит организация — соглашения о том, кто какие делает дела, каким образом координируются

работы, а не инженерные/технические практики сами по себе. То, из чего состоит мир (в том числе предприятие) обычно называют онтологией. Так что Ян Дитц (Jan Dietz) назвал свою книгу, объясняющую организацию с точки зрения теории речевых действий, “Enterprise Ontology”:




*Архитектура предприятия*

Jan Dietz не единственный, кто называет определение основных организационных решений онтологией предприятия (enterprise ontology). Основные решения, определяющее предприятие, называются архитектурой. Но компоненты, модули, размещения (и разные другие объекты) предприятия не так просто определить — поэтому люди обращают внимание на вопрос, “что это в нашем мире?”, т.е. от собственно архитектурного вопроса переходят к вопросу онтологическому.

Тут ещё нужно обратить внимание, что чаще в данном случае говорят про архитектуру предприятия (enterprise, у которого есть конкретный владелец, хотя иногда и коллективный — “акционеры”), и практически никогда про архитектуру предпринятия (endeavour, т.е. включающую и организацию отдельных проектов внутри предприятия, так и объединение нескольких предприятий для создания крупной системы). Но ничто не мешает определять архитектуру предпринятия по тем же принципам, что и архитектуру предприятия.

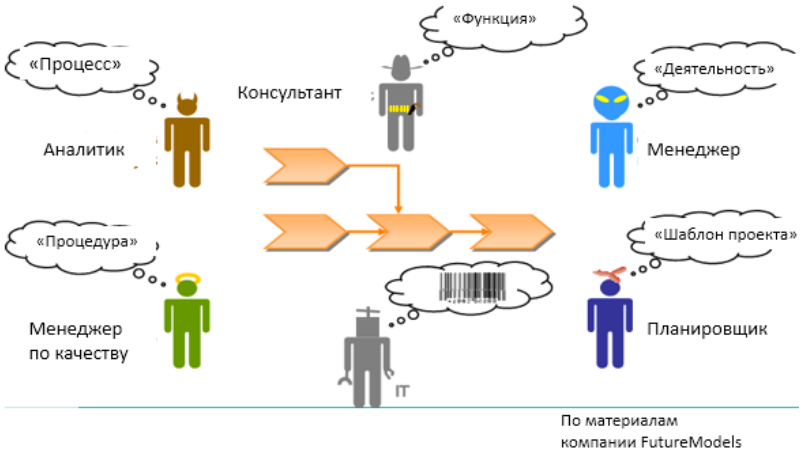
Архитектура предприятия ничем особенным не отличается от любой другой архитектуры: разве что у предприятия не совсем тривиальным образом выделяются компоненты, модули, размещения — ну, и принципиальные схемы предприятия (схемы деятельности) выглядят по-другому, нежели принципиальные схемы электрические и оптические.

Архитектура предприятия — это то, как оно организовано. Как организовано (архитектура) — это кто над чем работает, и кому эта работа нужна. Предприятие всегда как-то организовано, хорошо или плохо. Организация (онтология, архитектура) предприятия невидима, но можно сделать вполне видимое архитектурное описание. Если есть архитектурное описание, то можно его использовать для обсуждения организации предприятия всеми стейкхолдерами предприятия (включая компетентных в вопросах организации людей) — и тогда

есть шанс, что организацию удастся улучшить. Если документированного на каком-нибудь отчуждаемом из головы носителе информации архитектурного описания нет, то каждый имеет какое-то (не спрашивайте, какое) описание в каком-то (не спрашивайте, в каком) виде в собственной голове, и даже в ходе одного разговора это описание может у одного человека меняться три-четыре раза. При обсуждении организации работ предприятия у всех гарантированно будут разные представления о договорённостях, а результат работ по таким договорённостям описан в басне Крылова про лебедя, рака и щуку.

Архитектурное описание состоит из ряда диаграмм, по которым люди водят пальчиком, чтобы понять устройство организации и затем договориться, что нужно в организации поменять (обычно это верно и для любых других целевых систем: люди договариваются, что менять в системах, водя по архитектурным их диаграммам пальчиком). Архитектурное описание прежде всего — это инструмент для заключения "договорок" важных людей (заинтересованных сторон) по поводу важных аспектов организации работ. Не нужно путать организационные регламенты (в которых написано и важное, и не очень важное, и вообще много слов) с архитектурным описанием, в котором есть только самое-самое важное, зато выраженное максимально формально с целью избежать ошибок.

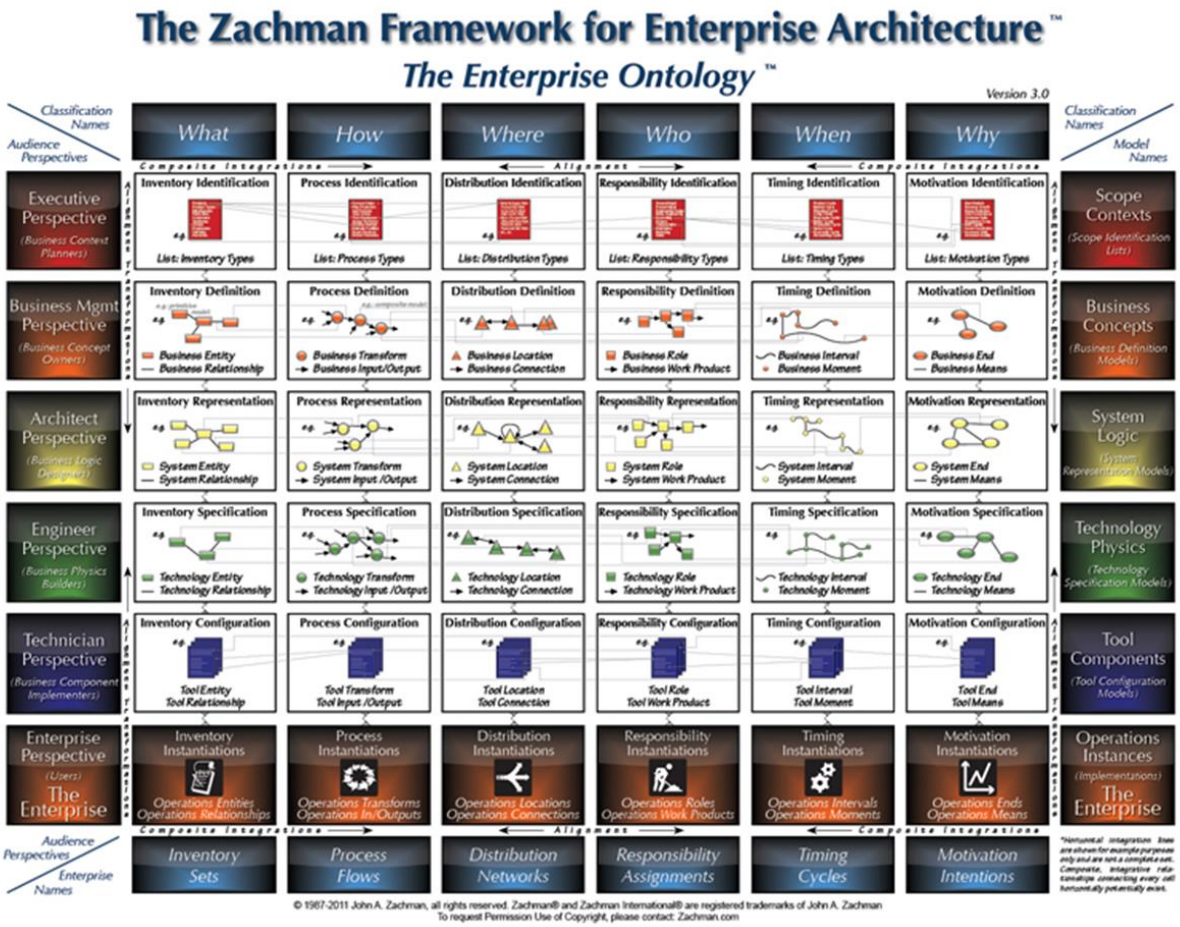
Опять же, большинство архитектурных диаграмм предприятия изображают какие-то процессы, процедуры, практики, проекты, функции, дела, деятельности, шаблоны проекта, экземпляры процессов, структуры подчинения. Нужно хорошо понимать, чем они все отличаются друг от друга, и что именно изображено на весьма и весьма похожих диаграммах. Мы возвращаемся к вопросу "что это" — говоря об объектах, из которых состоит деятельность, т.е. к онтологическому вопросу:




Подход Захмана к архитектуре предприятия

Одним из отцов-основателей архитектурного моделирования предприятий был John Zahman, который предложил подход к описанию архитектуры предприятия в виде матрицы. Он выпустил всего три версии такого описания. В третьей версии он ввёл подзаголовок "The Enterprise Ontology" (tm), подтверждая важность онтологического разбирательства с вопросом "из каких объектов состоит предприятие", плюс намеренно исключил все упоминания информационных технологий (даже упоминание данных предприятия) — с его точки зрения, определять предприятие должны не программисты и администраторы базы данных, а полномочные топ-менеджеры (и если просто упомянуть что-то айтишное, то

прибегут программисты и всё испортят: важный вопрос архитектуры будут обсуждать люди, у которых нет полномочий по воплощению организационных идей в жизнь).




Захман рассматривает какой-то жизненный цикл (от задумки до воплощения) элементов предприятия — reification transformation, и каждую стадию жизненного цикла связывает с определённой профессиональной ролью (стейкхолдером предприятия):

1. Scope context (identifications) — executive (business scope planners)
2. Business concepts (definitions) — business management (business concept owners)
3. System logic (representations) — architect (business logic designers)
4. Technology physics (specifications) — engineer (business logic builders)
5. Tool components (configurations) — technician (business components implementers)
6. Operations instances (instantiations) — users, The Enterprise (то есть уже индивид, а не класс).

Несмотря на то, что все модели захмановской архитектуры предприятия разного уровня абстракции (определение системы) и само предприятие (воплощение системы) наличествуют в конце концов во времени одновременно, можно условно сказать, что это неявная развертка по времени — указание на примерный жизненный цикл организации как системы, от задумки до организовывания, до функционирования. И приведённые шесть уровней воплощения — это неявно шесть стадий жизненного цикла предпринятия (или отдельной технологии предпринятия), каждая из которых имеет свой специфический набор моделей, свой cognitive framework/корпус знаний, своих озабоченных целостностью проводимых на данной

стадии работ стейкхолдеров (позиционеров-профессионалов).

Онтологически предприятие по Захману состоит из следующих объектов:

1. Активы (inventory sets) — сущности и отношения.

2. Ход работ (process flows) — трансформации, и что на входе/выходе этих трансформаций. Захман, конечно, представитель процессного подхода.

3. Распределительные сети (distribution networks) — это операционное управление, логистические сети, руление потоками.

4. Ответственность/полномочия (responsibility assignments) — это и есть "организация".

5. Временные циклы (Timing Cycles) — и обратите внимание, что "жизненный цикл" потихоньку уступает место "временнЫм циклам" (в том числе из-за давления путаницы с case life cycle и project life cycle — они-то уж точно не про "всю жизнь от замысла до смерти", речь не о жизни и смерти, а о кусочке времени в этой жизни). Захман тут следует общей тенденции выделять временнОй аспект деятельности как отдельный от описания практик (т.к. слово "жизненный цикл" закрепляется не столько за описанием стадийности, сколько за описанием деятельности в части используемых в ходе этой деятельности практик работы). Так, в стандарте ситуационной инженерии методов ISO 24744 (это предшественник OMG Essence) говорится: Stage-WithDuration/*Kind is, in turn, specialized into TimeCycle/*Kind (having as objective the delivery of a final product or service), Phase/*Kind (having as objective the transition between cognitive frameworks) and Build/*Kind (having as major objective the delivery of an incremented version of an already existing set of work products). TimeCycle/*Kind also holds a whole/part relationship to Stage/*Kind, allowing for the recursive composition of time cycles and other stages. Да, timing cycles и в ISO 24744 и у Захмана — это интервалы и моменты времени.

6. Деятельностное намерение (Motivation Intentions) — цели и средства, какие средства помогают достигать каких целей. Обратите внимание на то, что слово motivation обычно означает именно "мотивированное использование средств для достижения целей", речь не идёт о "мотивации" в смысле "как заинтересовать кого-то поработать".

Все эти объекты провязаны друг с другом через correspondence rules — как и предполагается в ISO 42010 (и слово rules тут не случайно. См. постинг http://www.ronross.info/blog/2011/09/06/where-are-your-business-rules-%E2%80%A6-in-a-big-p-process-dead-zone/ с разъяснениями — хотя я и понимаю, что business rules и correspondence rules далеко не одно и то же, но смысл использования каких-то логических пропозиций при обсуждении связи моделей из разных тематических групп описаний остаётся).

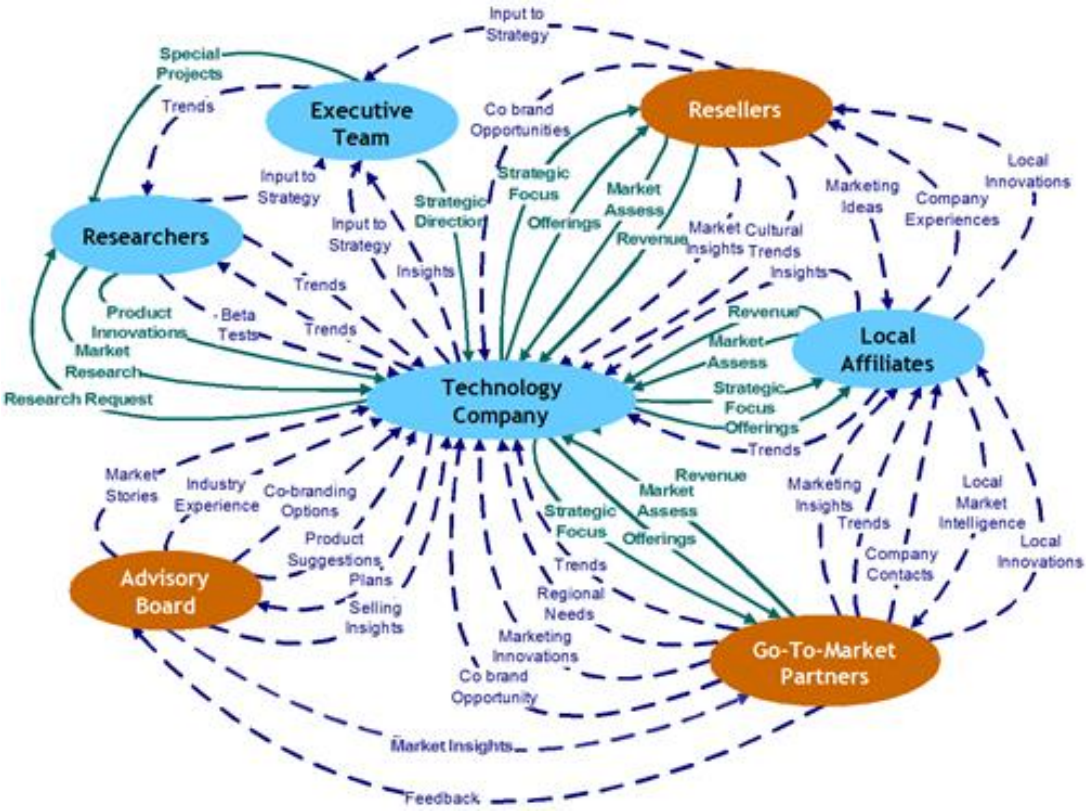
### Бизнес-архитектура

В архитектуре предприятия часто выделяют определение того, как предприятие будет получать возможности — бизнес-архитектуру. Есть два подхода:

- бизнес-архитектура — это просто часть общей архитектуры предприятия в целом.
- бизнес-архитектуры отдельна от любых других определений предприятия.

В бизнес-архитектуре особо выделяются описание сети пользы и бизнес-модели.

Сети пользы (value networks), позволяющие обсудить то, как добавляется польза от выстраивания каких-то цепочек обменов (value chain, value-added):




На основе согласованных между собой и объединённых восьми разных способов описания цепочек “добавления пользы” (ибо value не точно переводится как “стоимость”, т.е. “добавление стоимости” это слишком русско-марксистский перевод, а слово “ценность” уж слишком пафосно) был создан стандарт OMG VDML (Value Delivery Modeling Language): http://www.omg.org/spec/VDML/

Альтернативый язык описания бизнес-архитектуры это OMG BMM (business motivation model, http://www.omg.org/spec/BMM/) — на этом языке, как и обычно при использовании слова motivation, говорится о бизнес-целях (goals) их связях со средствами их достижения (means):

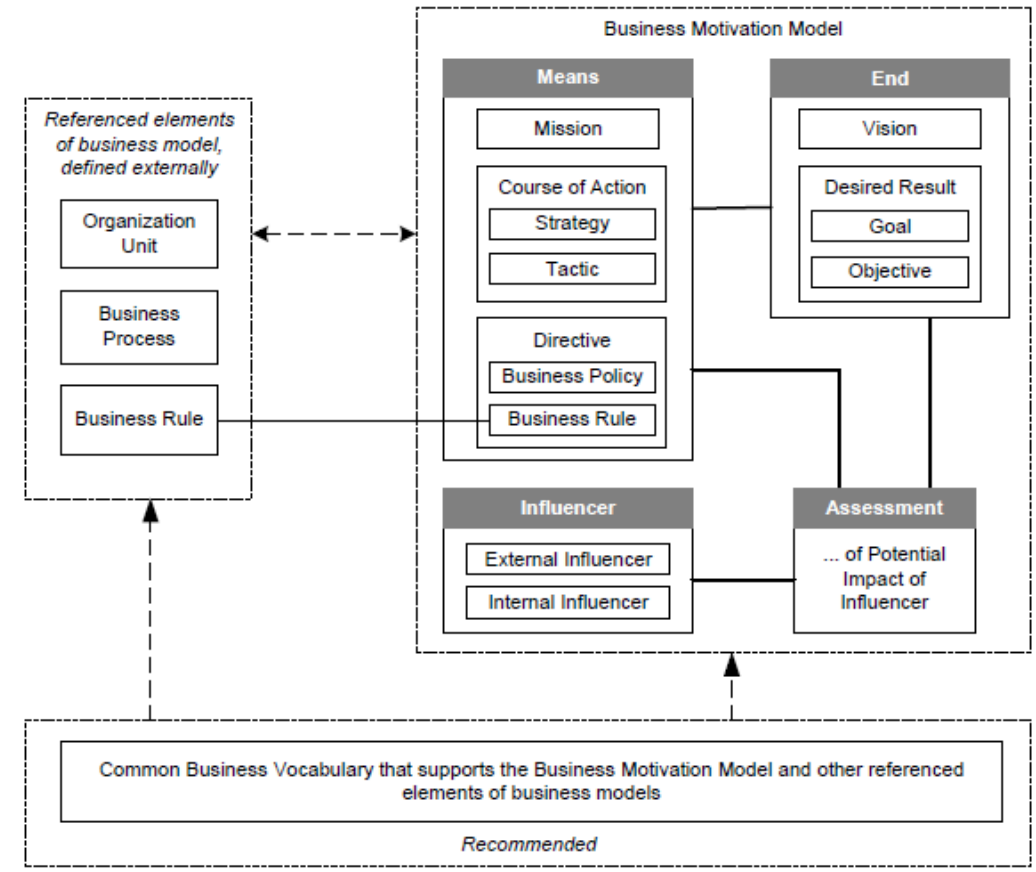
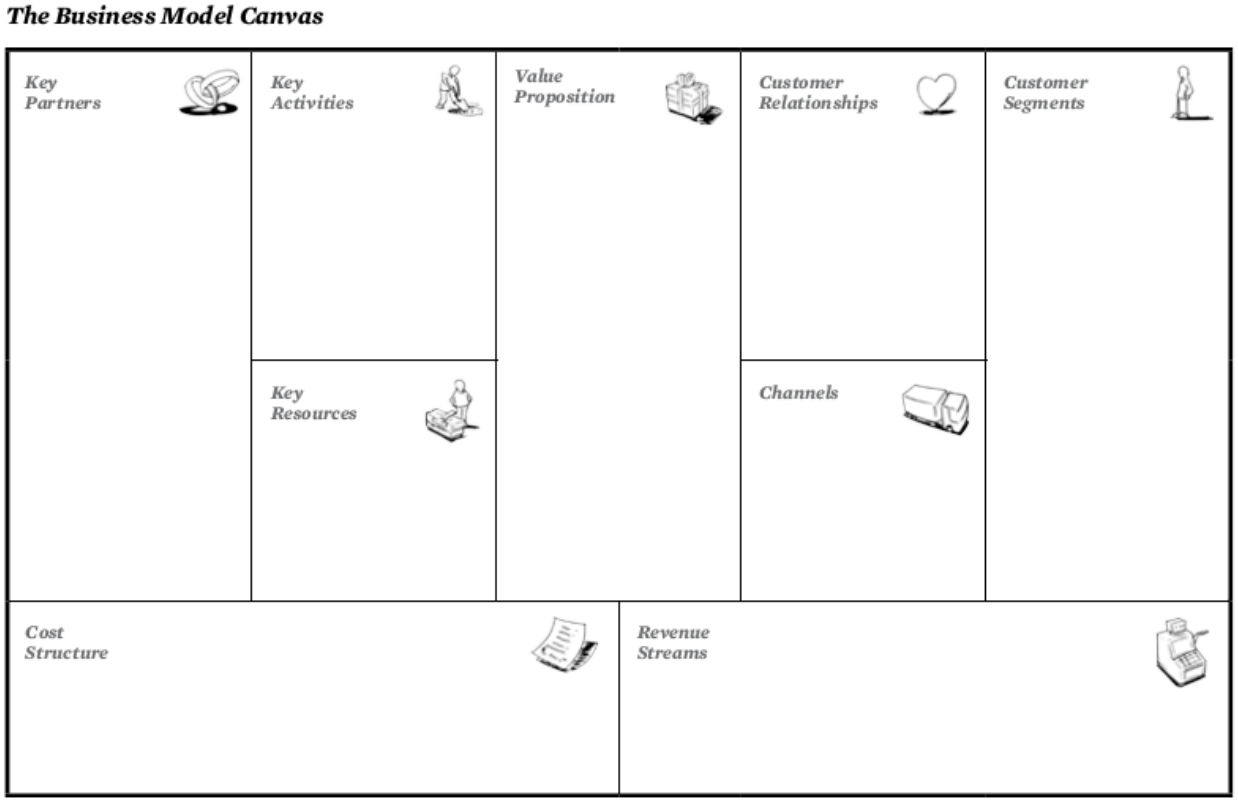



Figure 7.1 - BMM Overview

Помним, что каждое слово (mission, vision, goal, objective и т.д.) каждый стандарт, каждый язык определяет по-своему! Всегда выясняйте, что именно имеет ввиду тот или иной стейкхолдер, когда говорит о “миссии” или “цели” или даже о “бизнесе” (так, отнюдь не все, произносящие слово “бизнес-процесс” задумываются именно о бизнесе, “как заработать деньги”!).

Бизнес-модель — это ответ на вопрос, “как вы зарабатываете деньги на том, что делаете”. Есть множество разных способов описания бизнес-модели (от текстовых эссе до моделирования в VDML). Но сейчас набрал популярность способ описания при помощи “порождения бизнес-модели” (business model generation — http://www.businessmodelgeneration.com) — книга, которая предлагает описывать бизнес-модель через ключевые деятельности, ключевых партнёров, ключевые ресурсы, структуру затрат, отношений с клиентами, сегментирование клиентов, предложение пользы, каналы продаж, источники доходов — и для удобства изображать всё это на определённого типа диаграмме-”холсте” (canvas, http://www.businessmodelgeneration.com/downloads/businessmodelgeneration_preview.pdf, генератор .pdf файлов на разных языках -- http://www.bimoga.com/):




Бизнес-архитектура является традиционной вотчиной экспертов из McKinsey & Co, Bain & Co, Booz & Co, BCG, Palladium и прочих всеми ругаемых "консультантов не по делу". Именно это преподаётся в разных университетах в курсах MBA. Ключевые слова тут — "кому это нужно" (полезность, value), "справляемся ли мы с" (capability, над переводом думать и думать), "мы банда" (сообщество, кластер, фирма, рабочая группа) и т.д. Обратите внимание, как этот "предпринимательский" разговор (начинающийся с вопроса о том, кому всё это нужно и зачем мы всё это делаем) отличается от "работ/процессов", "акторов", "функций" и прочего привычного менеджерского и инженерного разговора, очень важного для организации эффективного производства (которое крайне важно, когда вы уже знаете, что производить, и вам нужно сделать подешевле и побыстрее). Кроме "побыстрее" и "подешевле", или даже "чтобы результат был качественный" (инженерный взгляд на вещи), бизнесменам нужно ответить на вопрос "зачем это вообще делать" (не технологическая причинно-следственная цепочка, а ценностная — нужна ли кому-нибудь затеваемая деятельность).

На сегодня есть:

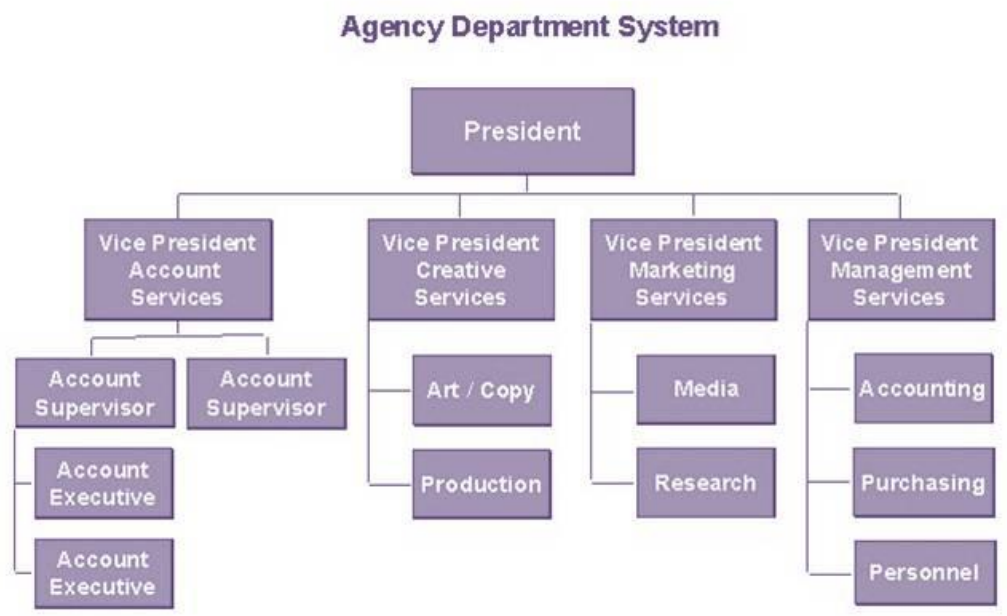
- гильдия бизнес-архитекторов — http://www.businessarchitectureguild.org/ (главный продукт — Business Architecture Body of Knowledge, BIZBOK™, уже версия 3.0). Это новое объединение, организации всего пара лет, они делают акцент на то, что это "знание архитекторов от сохи".
- ассоциация бизнес-архитекторов — http://www.businessarchitectsassociation.org/ (главный продукт — сертификация бизнес-архитекторов), это смычка университетов и консалтеров. Обратим внимание, что гильдия бизнес-архитекторов придерживается других взглядов на предмет, чем ассоциация (отчетливо это проявляется в репликах William Ulrich треда http://www.linkedin.com/groups/When-Stars-Align-Business-Architecture-4379346.S.130420677).

- SIG в OMG — http://bawg.omg.org/ (стандарт VDML — это как раз их рук дело, презентации http://bawg.omg.org/12-06-01.pdf)
- дискуссионная группа в LinkedIn — http://www.linkedin.com/groups/Business-Architecture-Perspectives-Transforming-Business-4379346
- разные вебсайты типа http://www.bainstitute.org/ (утверждают, что там комьюнити более 40тыс. членов, родственные сайты BPMinstitute.org и SOAinstitute.org — ну, вы поняли).
- школы типа http://paularthurbodine.com/the-chicago-school-of-business-architecture/
- … и много чего другого, ссылки наверху даже не надводная часть айсберга, это махонький кусочек.

Архитектура предприятия (включая тамошнюю архитектуру деятельности) — это то, чем сегодня занимаются не столько стратеги, сколько выходцы из айтишников, включая "бизнес-аналитиков" (которые опять же, про "деятельность" как тип для повторяющихся в чём-то дел, но не про "бизнес" в предпринимательском смысле этого слова), тоже вырастающих из “бывших программистов”. Это часть enterprise engineering, которая в свою очередь часть systems engineering в части систем предприятий. А вот бизнес-архитектура — это часть менеджерской/стратегической деятельности, предпринимательство, а не инженерия.

### Органиграмма

Органиграмма — это структура подразделений, декомпозиция предприятия, модульная его структура. Любую функцию предприятия должен кто-то поддержать: компоненты должны быть реализованы какими-то модулями. Если почту не отсылает канцелярия, то почту должна отсылать хозяйственная служба, или каждый работник сам. Как и любая архитектура, архитектура предпринятия считается созданной только тогда, когда вместе удалось совместить логическую архитектуру (требуемые деятельности — поведения, функции, процессы, практики, коммуникации) и физическую структуру, включающую обычно определение полномочий по распоряжению её физическими ресурсами. Традиционная “органиграмма” (organizational chart) как раз и выражает такую модель, при этом в ней обычно по понятным причинам перемешаны начальники и сотрудники (каждый начальник считается представляющим всех подчинённых ему сотрудников, он олицетворяет все его подразделения!):



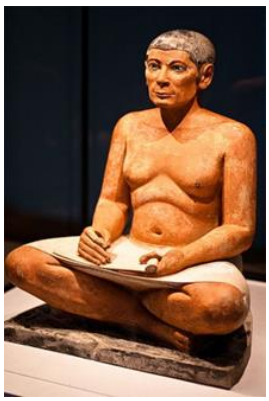

В этом примере из Википедии (Departments in advertising agencies, http://en.wikipedia.org/wiki/Organizational_chart#mediaviewer/File:Deprtments_in_advertising_agencies.jpg) это хорошо видно — и невпопад сказанное слово system (указание на то, что "над департментальным разбиением люди думали, оно неслучайно"), и отождествление всего агентства с президентом, и одновременное указание названия службы и главы этой службы ("vice president creative services") и название подразделений по их ведущим практикам (Research, Art/Copy). Это типично для органиграмм.

Много более запутанные структуры возникают при попытках отразить наличие множества совмещаемых между собой структур предпринятия-системы — например, гибридные диаграммы с показом соответствия функций и подразделений, а также диаграммы с показом наложенных друг на друга предпринятий (предпринятий-проектов и предпринятий-подразделений — так называемые "матричные" структуры).

Это всё требует специального рассмотрения, но понимание онтологической сути дела (множественности тематических описаний, иерархичности в каждом из описаний, физичности воплощения системы, отражённого в каждом из тематических описаний, отождествления темпоральных частей каждого из представлений в силу 4D extensinalism) позволяет довольно легко с этим разобраться.

### Писцы против инженеров

Помним, что инженер — это тот, кто преобразует информацию и вещество, а не тот, кто записывает чужие мысли. Писарь или писец своих мыслей не имеют, они пишут под диктовку, или записывают в порядке учёта то, что видят.



Архитекторы предпринятия — это те, кто определяют деятельность и информационные системы предпринятия, кто имеет полномочия их менять! Если же это отставные программисты, которые назвали себя “аналитиками” и лишь зарисовывают известными им иероглифами тайных архитектурных языков наговорённые им “старшими товарищами” или подсмотренные ими в жизни архитектуры, то это не архитекторы. Это живые диктофоны, машинописное бюро, даром что пишут они специальными значками в моделерах.

Интересно, что “инженер предпринятия” — нет такого понятия, хотя “архитектор предпринятия” есть.

### Неархитектурные описания предпринятия

Конечно, предпринятие моделируется довольно подробно в самых разных информационных системах — и отнюдь не все организационные, технологические, операционные/логистические описания являются архитектурными. Как и в случае архитектуры любой системы, а не только системы-предпринятия, отнесение описаний к архитектурным или неархитектурным зависит от решения архитектора: считает ли он, что при изменении этого решения придётся много чего переделывать-переорганизовывать-переосваивать в предпринятии, или же не так много. Что для одного покажется огромной работой и крайне важным (т.е. потянет на элемент архитектуры), другой выполнит “на автомате” и даже не заметит — и поэтому в архитектуру это не попадёт.

В предпринятии можно обнаружить самые разные описания деятельности — разного уровня детальности, для разных стейкхолдеров. Так, в ERP системе можно найти детальную (а не только архитектурную) информацию о том, какие именно ресурсы предпринятия наличествуют сейчас, и какие планируются на конкретную дату. В CRM-системе можно узнать не только архитектурные сведения про наличие важнейших стейкхолдеров, но и узнать совершенно неархитектурные данные о телефонах представителей каких-то групп стейкхолдеров.

Так, деятельность по «утренней продувке трубопровода низкого давления» можно обнаружить в крупном предприятии по-разному описанной минимум четырежды в самых разных подразделениях:

- в архитектуре предприятия, чтобы понимать, какие информационные системы её поддерживают, при этом архитектура предприятия часто будет разбита на два отдельно моделируемых куска:
    - «бизнес-процессы» и регламенты работы будут отмоделированы в "службе обеспечения качества" — ибо есть стойкое поверье, что описание процесса помогает именно качеству выполнения работ, и будет использован “процессный подход” (т.е. описана

последовательность шагов по продувке).

- архитектура информационных систем и аппаратного комплекса будут отмоделированы в «IT-службе» как IT-архитектура (а иногда и этот кусок бьют на отдельно архитектуру программного обеспечения и архитектуру компьютерного и связного оборудования).

- в описании метода/практики: что-то, близкое к описаниям ситуационной инженерии методов — какие рабочие продукты, инструменты, компетенции, отдельные операции нужно выполнять для обслуживания трубопровода низкого давления, для продувок трубопроводов и т.д.
- в обучающих "пакетах" (чаще всего в соответствии со стандартом SCORM) дистантного образования. Чаще всего это поддерживается HR-службой, именно в их задачах учить новых людей специфической для данного предпринятия деятельности.
- в системе документооборота, или системе процессного управления (BPM), или трекере ведения дел/выполнения поручений/документооборота (issue tracker, система adaptive case management), или системе проектного управления – для целей планирования работ и контроля их выполнения. Тут нужно обратить внимание, что все эти разные компьютерные системы позволяют планировать работы и отслеживать выполнение работ, опираясь на разные подходы к их описанию: issue tracker чаще всего связывается с "управлением делами" (case management) как одним из ярко выраженных представителей информационно-ориентированного/опирающегося на рабочие продукты подхода и не подразумевает прописывание цепочек операций, а вот система проектного управления чётко предпишет место и время этой работы среди других работ (это вариант "процессного подхода").

Это всё системный подход

Все эти архитектурные и неархитектурные описания предпринятия, выраженные в текстах, компьютерных моделях/базах данных, регламентах, стандартах, приказах, протоколах, компьютерных программах (в коде программ, шаблонах отчётов баз данных и т.д.) представляют собой описания совместной деятельности людей, зачастую включая описания взаимодействия людей с информационными системами и производственным оборудованием и инструментами, а также описания работы информационных систем и производственного оборудования и инструментов. Эти разные описания удовлетворяют интересы разных заинтересованных сторон (stakeholders) предпринятия путём задействования разных тематических методов описания (viewpoints) для создания разных тематических групп описаний (views), включающих в себя отдельные более и менее формальные модели — разве что методы описания предпринятия/деятельности не похожи методы описания для "железных" или "программных" систем. Но вот то, как и что описывается в предпринятии-системе можно обсуждать примерно так же, как это можно обсуждать в отношении любой другой (программной или "железной") системы. Это и есть мощь системноинженерного подхода: экономия мышления.

Всё, что говорилось про определение системы, оказывается верным и по отношению к предпринятию — например, необходимость управления конфигурацией/изменениями. Самые разные модели должны отражать одну и ту же планируемую и актуальную деятельность, версии разных моделей должны

соответствовать друг другу. САПР “инженера предпринятия” (организатора) — это различные моделеры языков описания деятельности.

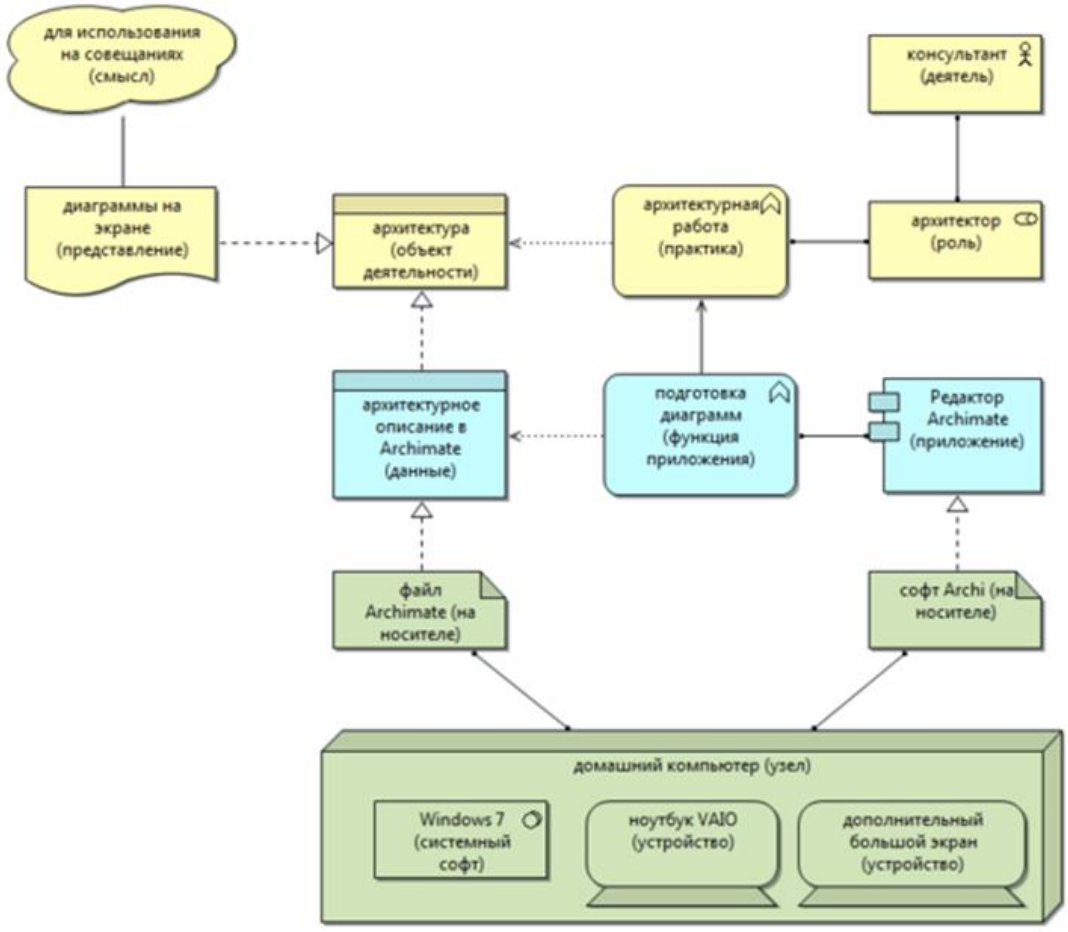
### *ArchiMate*

Для описания архитектуры предприятий разработано много подходов (frameworks), определяющих различные наборы viewpoints — TOGAF, Zachman’s, ArchiMate. Также определено какое-то количество архитектурных языков, которые предназначены для выражения идей, содержащихся в подходах к моделированию предприятий (из которых на сегодняшний день выделяется стандарт OpenGroup ArchiMate 2.0 — его полный текст http://pubs.opengroup.org/architecture/archimate2-doc/, набор ссылок на русскоязычные тексты по Архимейту: http://ailev.livejournal.com/988360.html).

Основная дизайн-цель при создании Архимейта была учесть необходимость указания в архитектуре предприятия одновременно:

- Структур деятельности (людей)
- Структур функционирования компьютерных программ
- Структур поддержки инфраструктуры (компьютеров и линий связи) для программ.

Для этого был придуман язык, на одной и той же диаграмме которого можно было отразить объекты и связи всех этих разных категорий. Вот пример того, как изображаюсь я, занимающийся архитектурой предприятия своего клиента в редакторе Archi (свободный редактор для софта Archimate):




Жёлтым выделена структура деятельности людей (консультант в роли архитектора, занимающийся архитектурной практикой), голубым — работа программ, зелёным —

оборудование, на котором работают программы. Все эти три уровня показаны на одной диаграмме, это и было главным достижением предложенного подхода и языка.

В 21 веке предприятия координируются уже не столько непосредственной коммуникацией людей, которые держат основной объем координационной информации в головах, сколько опосредованной компьютерами коммуникацией и хранящейся в компьютерах координационной информацией. Архимейт (как и вся дисциплина архитектуры предприятия) учитывает этот факт — но Архимейт стимулирует архитекторов не забывать о том, что кроме компьютеров и программ на предприятии есть ещё и люди (удивительно, но чаще архитекторы предприятия забывают про людей, прописывая подробно решения IT-архитектуры, а не забывают про IT-архитектуру, прописывая подробно деятельность людей. Якобы "архитекторы предприятия", а на самом деле "айтишники" иногда забываются до того, что "архитектура предприятия" воспринимается ими чуть ли не как синоним "архитектуры IT-решения". Это неправильно!).

Далее рассказ про Архимейт будет намеренно упрощён и в нём будет максимально использована неспецифическая терминология. В качестве упражнения попробуйте рассказать о всём том же самом с использованием понятийного аппарата настоящего курса (и заодно обратите внимание: все терминологические сложности настоящего курса системноинженерного мышления вполне можно обходить, если обращаться к определённой аудитории — но при этом сразу будет теряться общность мышления о самых разных системах!).

### Зачем нужен Архимейт

Моделирование на Архимейте — это создание таких диаграмм, при помощи которых мы можем что-то узнавать про предприятие. Архимейт имеет встроенные в него способы извлечения знаний о предприятии — он заставляет архитектора задавать вопросы (в том числе и самому себе) там, где есть неочевидные сразу "непонятки". Ответы на вопросы чаще всего требуют либо изобрести чего-нибудь (если речь идет о проектировании нового предприятия), либо разузнать чего-нибудь (если описывается уже имеющееся предприятие), либо просто исправить очевидную ошибку.

Архитектор не гроссмейстер — он не умеет играть без доски, он использует диаграммы, чтобы смотреть на них и думать. Тем самым работа его двухтактна: писать диаграммы и затем читать их. Эти диаграммы для него — "экзокортекс", продолжение его мозга, участвующее в размышлении. Он думает, ставя перед собой (или перед другими — если организуемое им размышление коллективное) вопросы, появляющиеся от соприкосновения формализма Архимейта с неформально устроенной жизнью предприятия. Тут неважно, это только проектируемая будущая жизнь, или реальная уже идущая жизнь (всё одно мы и об одной, и о другой узнаём только то, что думают об этой жизни люди, а не то, что было или есть "на самом деле"). Мысль — диаграмма — вопрос — мысль.

Главным механизмом Архимейта, стимулирующим задание вопросов, является механизм типов для объектов и отношений. Например, для процессов вы можете указать связь по передаче информации (но ничего не сказано про последовательность исполнения), или по запуску (подразумевающему указание последовательности выполнения) — и у вас будет много вопросов про сами процессы, как только вы начнёте выбирать тип их связи в отношении. Если у вас два объекта деятельности, и вы хотите указать какое-то отношение между ними, а

Архимейт предлагает из подходящих типов только "связь" без дальнейшей спецификации ("связь без причины — признак дурачины"), то вам нужно попробовать поискать в жизни какое-то поведение, связывающее эти объекты — и отмоделировать это поведение явно на диаграмме. Нужно запомнить: "за отношением обычно стоит глагол, а за объектом — существительное", и попытаться отыскать этот "глагол" (работу) в жизни.

Механизм типов подразумевает задание главным образом одного простого вопроса, ответ на который обычно очень сложно получить. Для каждого встречающегося в жизни объекта спросите "что это?!", "часть чего это?", "с чем это связано?". Подберите тип Архимейта. Получите удовольствие: до вас мало кто задумывался, "что это". Ответ на этот вопрос может оказаться очень нетривиальным, а получение этого ответа заставит задать десятки других вопросов.

Вторым механизмом является предписанное именование. Если у вас практика — то это отглагольное существительное, например, "полевой инжиниринг" (а хоть эта "отглагольность" и идёт тут от английского). Существительные плохо ассоциируются с "последовательностью выполнения", кооперативными цепочками работ разных людей. А вот процессы — это глагол, и придётся написать "инжинирить в поле". Вы этого хотели? Если нет — то у вас действительно процесс, вы уверены? Если правильный вопрос к объекту не "что делать?" — то придется признать, что речь идет не о процессе, а о чём-то другом (практике? роли? — это все "промежуточные" типы между чистой работой и чистым выполнителем).

Третьим механизмом является формализм: следование диаграмм логическим правилам (типа "часть части входит в целое" для отношений состава). Формальное можно проверить на непротиворечивость, а потом сравнить результат с жизнью: если в жизни обнаруживается противоречие там, где его нет на диаграмме, то нужно искать причины этого противоречия. Так, ваши "производственные задания" входят в "план сооружения". Ещё через несколько минут вы записываете, что "план сооружения" входит в "проект сложного инженерного объекта". И тут, глядючи на эти два появившихся в разное время на диаграмме отношения композиции, вы вдруг понимаете, что на диаграмме "задания" в "проект" входят, а вот в жизни "задания" в "проект" обычно не входят. Нужно думать, почему так вообще написалось. Скорее всего, вас проблемы с тем, что вы называете "планом": возможно, что планов два — один (предварительный) входит в проект, а второй (оперативный) не входит в проект, а задания как раз входят в него. Или где-то имелась в виду модель, а где-то выписка из модели — и эта выписка или модель начали собственную жизнь после момента получения выписки. Или ещё что-то: нужно идти в жизнь, и исследовать — а не гадать. А потом исправлять диаграмму.

Моделирование идёт циклически: поглядел на жизнь и что-то про неё понял, затем записал понятое на диаграмме. Поглядел на диаграмму — нашёл ошибки. Поглядел на жизнь — исправил ошибки в диаграмме. Поглядел на диаграмму — понял что-то про жизнь. Записал понятое на диаграмме — и опять начал искать ошибки. Результат такого процесса довольно неожиданный: архитектор вдруг начинает что-то понимать про жизнь предприятия едва ли не глубже, чем непосредственно участвующие в деятельности люди. Архитектор находит в действующих предприятиях бессмысленные работы (activity, кстати, рекомендую переводить как "работы" — родовой термин для всех вариантов поведений), странные группировки объектов деятельности, лишних деятелей. Он же находит их в проектах предприятий — как своих проектах, так и в проектах других людей. Формализм Архимейта заставляет продумывать то, что прошло бы мимо внимания. Ошибки и

трудности моделирования являются теми ниточками, с которых можно начинать распутывание запутанного клубка деятельности предприятия — и Архимейт в изобилии поставляет такие ниточки.

Особенностью моделирования на Архимейте является его кажущаяся аскетичность. Моделирование делается намеренно неподробно (при всех заявлениях, что "при потребности вы сможете выразить все нужные детали). Типов объектов и отношений в Архимейте намеренно мало, авторы заявляют что сознательно пошли на удовлетворение только 20% потребностей архитекторов в выразительных средствах (эти 20% покрывают 80% всех случаев, а остальные 80% "выразительности" ушли бы на оставшиеся 20% случаев — эти "остальные выразительности" просто убрали из языка). Но вся эта неподробность и аскетичность кажущаяся. Архимейт заставляет писать суть дела — как бы банально и просто эта суть ни выглядела на диаграмме. И — чудо, чудо — эта банальная суть дела всегда вызывает много банальных вопросов как к жизни, так и к диаграмме. Архимейт создан так, чтобы задавать правильные вопросы. Ответы на эти вопросы почему-то оказываются небанальными, и в этом главная сила Архимейта.

Обратите внимание, я тут удержался, и ничего не говорил про "онтологию". Но для тех, кому это слово не чуждо, я таки скажу: типы Архимейта таки представляют собой онтологию предприятия — понимание авторов Архимейта того, что можно найти на любом предприятии. Это означает, что на любом предприятии можно найти объекты деятельности и деятелей, процессы и данные, функционал программ и узлы IT-оборудования. Архимейт заставляет находить в окружающем мире именно эти (а не какие-то другие — например, "трансакции DEMO") объекты и отношения. Именно задание этой формальной онтологии и порождает вопросы, прежде всего классический онтологический вопрос про обнаруженный в жизни объект: "что это?!".

Формальные диаграммы из типизированных объектов и отношений и предписанные виды имён направляют архитектурное мышление, они ведут его по рельсам (да, так жестко!), предусмотренным авторами Архимейта. Это не так плохо, ибо если никак не ограничивать мышление архитектора, оно будет крайне нетехнологично, т.е. хорошие результаты не будут предсказуемо повторяться: один раз из ста у одного архитектора вдруг получится гениальный результат в сто раз лучше, чем с использованием Архимейта (обычно такие гении как раз и создают Архимейты и другие архитектурные языки), а девяносто девять раз у девяносто девяти архитекторов из ста получатся результаты сильно хуже. "Неограниченные Архимейтом архитекторы" в меру своей гениальности найдут на предприятиях "организационный флогистон", отношение "духоподъёма" между людьми, да и вместо "людей" неожиданно могут обнаружить какое-нибудь "быдло" или "ангелоидов". Плюс нужно будет для этих "флогистона" и "ангелоидов" придумывать графическую нотацию. Архимейт даёт специальные понятийные очки, через которые предприятие видится исключительно предписанным авторами этого стандарта способом. Способ это современен: сервисы, трехуровневость реализационного описания; различение объектов работы, работ и выполнителей (людей, программ, оборудования); и т.д. Это всё контринтуитивно, и не соответствует интуиции людей, хорошо знакомых с предприятиями и хорошо знакомых с программированием. Контринтуитивность предлагаемого Архимейтом взгляда на предприятие порождает много вопросов — неформальное предприятие не укладывается в новомодные формулы. Но формальность диаграмм позволяет их хоть как-то проверять, а также соотносить с жизнью.

Удобная графическая нотация является при этом только бонусом, секрет успеха вовсе не в ней. Секрет именно в типах элементов и отношений Архимейта, предлагаемых соглашениях об именовании и декларируемом соответствии формальных моделей Архимейта замыслу предприятия или реальному предприятию — и тех вопросах, которые эти типы, имена и отражение реальности моделью задают архитектору.

### Люди, программы, оборудование

Самым-самым важным в предприятии Архимейт считает наличие трёх уровней работ, на каждом из которых уменьшается человеческое начало: людей, программ и оборудования. Люди без программ беспомощны, программы без оборудования мертвы. Оборудование без работы программ — бесполезный кусок железа, программы без работы людей тоже не нужны. Так что в архитектуре предприятия обязаны быть представлены все три уровня выполнения работ в их взаимосвязи.

На каждом уровне есть свои выполнители работ и свои объекты работы. Собственно, работа заключается в том, что выполнители изменяют как-то объекты работ. Выполнители работ и объекты работ обычно представлены существительными, работы — глаголами и отглагольными существительными. Важно, что объекты работ сами ничего выполнять не умеют, они пассивны. А вот выполнители активны, они-то и трудятся над объектами и с использованием объектов.

Уровень людей содержательный. Люди за информацией видят те объекты окружающего мира, которые эта информация изображает. Смотрят на прогноз погоды и видят завтрашнюю погоду (а не описание погоды), смотрят на отчёт о стройке и видят количество реальных этажей (а не собственно отчёт), смотрят на отчёт о прибылях и убытках и видят ту самую прибыль. У людей есть цели, полномочия (могут выдавать некоторым другим людям поручения на выполнение работ) и ответственность (должны обещать, что выполнят поручения некоторых других людей). Целенаправленная деятельность есть только на этом уровне.

Уровень программ — это обработка информации, заключенной в данных. Из одних данных программы делают другие данные, отличающиеся как форматом, так и содержанием. Никто никому ничего не обещает (обещать программы не могут, это могут только люди) и не даёт поручений (поручать могут только люди), не преследует никаких целей (цели есть только у людей). На этом уровне известно, что означают данные в реальном мире: ведь опасно к килограммам прибавлять километры. Главная задача уровня программ — чтобы нужным способом обработанные данные оказались в нужный момент у нужных людей.

Уровень оборудования — это бездушный мир, в котором никакой обработки данных уже нет, а есть только хранение и пересылка данных. Конечно, на уровне оборудования тоже есть программы, но они уже другого рода — тут уже никто не знает, что означают эти данные в реальном мире. Задача оборудования — хранить адресуемые как-то байты, не вдаваясь в их смысл, пересылать эти байты по запросам программ, а также хранить сами программы и давать им возможность выполняться.

### Элементы и отношения

Предприятие в Архимейте описывается в виде элементов (изображаются разными фигурами), находящихся друг с другом в каких-то отношениях (отношения изображаются в виде соединительных линий между фигурками элементов).

Архимейт ценен тем, что предлагает для описания работы предприятия всего

- 16 типов элементов для уровня людей,
- 7 типов элементов для уровня программ,
- 9 типов элементов для уровня оборудования,
- 11 типов отношений, в которых элементы могут находиться друг с другом, и показ развилок для этих отношений.

Если вы собираетесь как-то менять архитектуру предприятия в реальной жизни (а иначе зачем вы вообще стали рисовать диаграммы Архимейта?), то для этого можно использовать ещё:

- 7 типов элементов для целеполагания и обоснования изменений в организации
- 4 типа элементов для проектирования перехода к новой архитектуре

Ещё есть "комментарий" и отношение связи "комментария" с какими-то другими элементами, а также рамочка для группировки элементов.

Вот и весь Архимейт. Но не нужно обольщаться его простотой. В Великом и Могучем тоже всего 33 буквы.

Нужен не ты, нужен твой сервис.

Важно, что никакие работы на предприятии не делаются просто так, они все кому-то зачем-то нужны. Сервисы — это полезные для каких-то других выполнителей (точнее, полезные для работ этих выполнителей) работы. Для потребителей сервисов абсолютно неважно, как организовано выполнение этих работ: кто с чем работает, чтобы выставить сервис для внешнего потребления. Для них важно только, по каким каналам (электронная почта, окошечко с девушкой, телефонный звонок и т.д.) и интерфейсам (если это программы) будут предоставлены эти сервисы.

Есть сервисы оборудования, предоставляемые программам, и сервисы программ, предоставляемые людям. Три уровня предприятия склеены именно этими сервисами — из каждого уровня главным образом видны только сервисы других уровней.

То есть можно заменить всё оборудование, и программы этого не заметят, если сервисы оборудования остались прежними. Так же и с программами: замени их все, но если сервисы останутся теми же, то люди это вполне переживут. В принципе, это относится и к самому предприятию: если сервисы людей, которые предприятие оказывает окружающему миру (а объединение таких сервисов и связывающего их контракта называется продуктом) будут выполняться совсем по-другому организованными людьми, которые пользуются совсем другими программами, которые работают на совсем другом оборудовании, то клиенты этого не заметят. Этим и пользуются архитекторы предприятия: они описывают предприятие, а потом потихоньку меняют программы и оборудование, поддерживая предоставление критически важных сервисов. Это и называется сервис-ориентированным подходом: разделять (разные уровни работы сервисами) и властвовать. Так что это ещё как посмотреть: предприятие склеено сервисами, а для архитектора оно этими сервисами разделено.

Стремление разделять и властвовать у архитекторов так велико, что они разделяют

сервисами и работы даже одного и того же уровня. Например, легко представить себе программы, оказывающие сервис не людям, а другим программам. Или оборудование, смысл существования которого — служить (оказывать сервис, т.е. "работать для") другому оборудованию.

Для предоставления внешне видимой работы-сервиса нужно выполнить много-много извне невидимой внутренней работы — изменения объектов работы выполнителями работ. Наличие этой границы внутреннего и внешнего рассмотрения ("черного ящика" с невидимыми внутренними выполнителями, работами и объектами против "прозрачного ящика", когда они отлично видны) — это наличие границы системы. Архимейт моделирует системы, разделяя части/уровни предприятия сервисами (хотя про "системы" в спецификации Архимейта и не сказано ни слова).

### Люди

Напомним пункт три: для описания уровня организации работы людей Архимейт имеет столько же типов элементов (17), сколько для уровней программ и оборудования вместе взятых (7+9) — и это совсем не случайно.

Сами люди в Архимейте представлены не своими личностями. В Архимейте не человек красит место, а место красит человека. В Архимейте выполнителем является живой человек, но только занимающий организационное место. Поэтому Архимейт "людьми" называет эти самые места, кто бы их в данный момент ни заполнял — один человек, целая группа людей из организационного звена (от сектора и отдела до филиала в другом государстве или даже всей компании целиком), временные работники, случайные граждане-клиенты, партнерские фирмы. Архимейт понимает, что это все эти оргместа занимают живые люди, но называет этих людей именно по наименованиям оргмест. Разнорабочий, зам.нач.отдела доставки, отдел доставки, филиал в Мытищах, клиент, аудитор — это и есть "люди", кто бы ни были те сотрудники, которым доведется занимать эти организационные позиции. Архитектор, как водится, заботится о вечном: если разнорабочий или президент заболели, и один человек подменил другого на время болезни, диаграмма останется верна: организация труда не изменится.

Люди Архимейта более разнообразны, чем те люди, которых можно найти в "оргструктуре" (органиграмме): отнюдь не все отношения между людьми Архимейта сводятся к начальствованию-подчинению. Так, партнёры и клиенты обычно в оргструктуре не упоминаются, а в Архимейте их показ — обычное дело.

### Роли

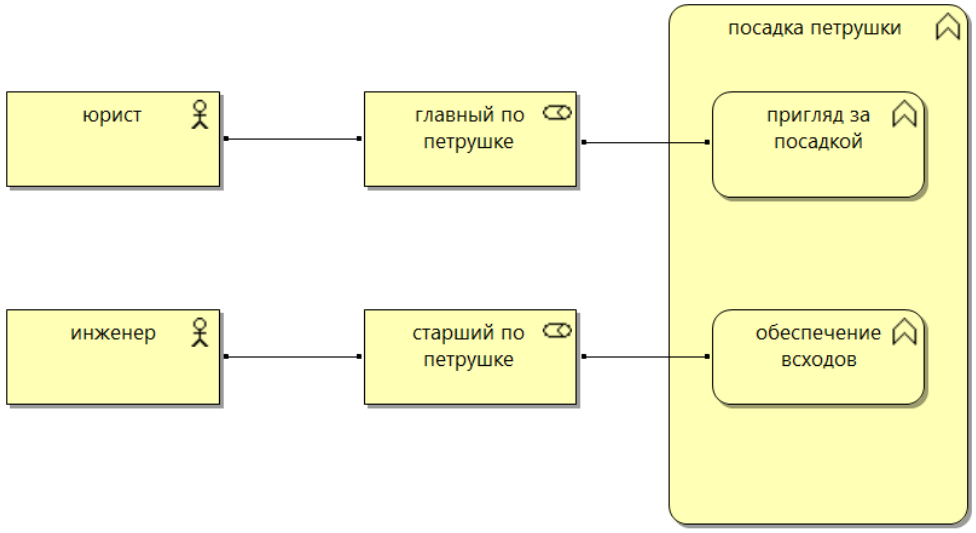
Архитекторы настолько коварны, что работают и с частями людей, называя их "роли". Ролью называется та выделенная во времени часть людей, которая выполняет определенный ряд работ, требующих какой-то особой квалификации. Так, если разнорабочий в театре вынужден то петь, то танцевать, у него есть две роли: когда он поёт, то роль — певец, а когда танцует — роль танцор.

Люди назначаются на роли, а роли назначаются на работы. Особо нужно отметить факт, что архитекторы организации занимаются организацией работ, а не лидерством (leadership). Распределение людей по ролям, а ролей по работам — это забота архитектора организации. А забота лидера — это а) назначение на места "людей" живых "человеков" с фамилиями, именами и отчествами, вредными и полезными привычками, определенными умениями и б) убалтывание кнутом и пряником человеков на местах "людей" выполнять назначенные им работы. Так что

фамилий на диаграммах Архимейта не увидишь: только должности, роли, подразделения, фирмы, "люди при исполнении", "представители", коллегиальные органы, временные группы и объединения и т.д.

В жизни назначение на роли и на работы обычно отражается приказами, распоряжениями, положениями и прочими "распорядительными документами". Такая система дробления людей нужна не только, чтобы показать многогранность занятий людей (напомню: и отдельных людей, и целых подразделений — когда большая компания из 1000 сотрудников выступает как в роли поставщика по отношению к одним фирмам, так и в роли заказчика по отношению к другим фирмам), но и чтобы свои диаграммы архитекторы меняли поменьше, а устанавливающие организацию работ приказы издавались пореже.

Типичный пример: в организации начинают заниматься посадкой петрушки, но пока непонятно, кто будет эту петрушку сажать — начальство говорит, "вы покажите, что там нужно будет делать, а я уж тогда назначу подходящую кандидатуру". Пишут "Положение о петрушке", где вводят роль "главного по петрушке" и "старшего по петрушке" и прописывают все их работы. Заметим, что Положение — это ещё не приказ, и не распоряжение. Это констатация факта, что какие-то организационные места (главного и старшего по петрушке) нужно заполнить, чтобы работы (по посадке петрушки) пошли. Начальник изучает этот проект положения, и заключает, что посадкой петрушки у него должны заниматься инженеры и юристы. Далее он пишет приказ, в котором он а) утверждает положение и б) прописывает, что роль главного по петрушке будет занимать юрист, а старшего по петрушке — инженер. Вот как это выглядит на архимейт-диаграмме:




Соединительные линии с точечками — это и есть "отношение назначения" между элементами. Отношения есть, конечно, и между людьми и работами непосредственно (если опустить упоминание роли). Такое отношение будет называться производным, но оно всё равно есть — можно говорить, что в данном случае инженер назначен на обеспечение всходов, а юрист на пригляд за посадкой.

Обратите внимание, что эта конструкция будет работать, даже когда инженер Иванов уйдёт на пенсию, а вместо него на это место будет принят Сидоров. В этой конструкции легко поменять инженера на агронома, или на Дирекцию по петрушке (когда деятельность разрастётся) — этот механизм назначения ролей на работы (например, Положением), а людей на роли (например, Распоряжением) работает и в случае, когда "люди" — это целая толпа людей, т.п. речь идёт об организационном звене. Все эти Положения и Приказы на диаграмме обычно не отмечаются: на

диаграммах договариваются, как будут организованы работы, а не какими документами эти договоры будут оформлены.

Подсказка: я знаю несколько организаций, где альбомы архитектурных описаний предприятия утверждались директором вместо толстой пачки регламентов. Ибо архитектурные диаграммы оказывались точнее и выразительнее многочисленных текстовых описаний, которые потом пытались по ним составить.

Понятно, что на посадке петрушки в конечном итоге работают самые живые Иванов и Сидоров (или сотрудники, которые числятся в Дирекции по петрушке). Но на предприятие они будут тратить только часть своего времени и талантов (и только эта часть времени отражается элементом "люди" в архитектуре предприятия), а остальное время спать, есть, гулять, учиться и развлекаться. На их роль в работах по посадке петрушки они будут тратить только часть времени, которое потратят на предприятие — ибо у них наверняка много разных ролей, они выполняют много разных работ. Да и на одну и ту же роль вполне могут быть назначено и некоторое количество людей — отдельных человек или даже несколько подразделений.

И людей, и ролей можно дробить на любое число частей, если нужно показать какие-то детали занятости людей-выполнителей.

Для выполнения какой-то работы разные люди могут на время объединиться (это обычно никак не отражается в оргструктуре), и такое временное объединение называют коллегиальной ролью.

### Работы людей

Работы людей бывают такие:

- уже случившиеся — события. Называют обычно события глаголом совершенной формы прошедшего времени: "петрушка посажена". Эти работы могут быть выполнены непреднамеренно ("случилась ошибка"), людьми не из предприятия (клиентами, конкурентами, партнёрами), да и не вообще эта работа могла быть выполнена не людьми (а, например, программами, оборудованием, или даже силами природы — "солнце зашло").
- направленные на достижение результата, и исполняемые в развёртке времени (часто — друг за дружкой) — процессы. Их называют глаголом в неопределённой форме — "копать", "посадить", "разработать". Процессы обычно запускаются событием, и запускают события, а также запускают друг друга, образуя цепочку от начального события до события-результата.
- полученные в результате работы временного коллектива, объединённого коллегиальной ролью — коллегиальные процессы. Их тоже называют глаголом в неопределённой форме.
- выделенные по какому-то иному критерию, чем развёртка во времени для получения результата (например, требующие назначения на них ролей с определённой квалификацией или потребляющие какой-то специфичный вид ресурсов) — практики. Практики не столько сами выполняются, сколько в разные моменты выполняются те или иные объединённые ими процессы (или фрагменты процессов, до которых дело на диаграмме так и не дошло, поэтому их изобразили без развёртки во времени, чохом, "в мешке" — то есть обозначили "что практикуется", а не указали на выполнение в какое-то определённое время). Поэтому практики обозначаются не глаголами, а отглагольными существительными: "посадка петрушки" это именно практика.

- предоставляемые кому-то вовне работы, как они значимы и видятся снаружи — сервисы. Сервисы реализованы внутренними работами, выполняемыми внутренними ролями, и используется в ходе внешних работ, выполняемых внешними ролями. В принципе, любые работы людей выполняются только для того, чтобы реализовать какой-то сервис — то есть для того, чтобы кто-то другой (например, клиенты или смежники) смогли воспользоваться этими работами.

В принципе, все предприятие разбито на какие-то части, и эти части выставляют наружу (для других частей предприятия, или за границы предприятия, или для другого уровня) сервисы, чтобы оправдать своё существование. Если есть трудности в понимании, кто в каких работах использует те или иные сервисы, или какие сервисы реализуют те или иные работы — значит, вы что-то не знаете, или не продумали.

Для уточнения того, чем же именно ценен сервис, в Архимейте даже есть специальный элемент: внешняя польза, которая связана с сервисом.

### Архитектура IT-решения

Компьютерные программы и оборудование (компьютеры и линии связи, в том числе "системные программы" — операционные системы, утилиты, программы ведения баз данных и т.д., не связанные с поддержкой данных предметной области предприятия) обычно называют IT-решением (IT-solution).

В современном предприятии ввиду глубокого разделения труда люди, обслуживающие оборудование ("железо") и прикладные программы находятся часто в разных подразделениях, более того: часто и люди, обслуживающие разные программы тоже находятся в разных подразделениях — и поэтому все "тянут одеяло в свои стороны", когда речь заходит о развитии предприятия. Архимейт-диаграммы позволяют обсудить спорные вопросы, и выбрать устраивающее всех решение — за счёт различения функционала программ (функциональных объектов, компонент) и модулей (собственно программных модулей). Так, если базы данных поддерживаются различными модулями в корпоративных информационных системах, то функционал хранения данных может быть отнесён к различным модулям — и это решение можно откладывать "на потом", сначала определяя на диаграммах потребный функционал (т.е. логическую архитектуру IT-решения), а уж потом занимаясь физической архитектурой отнесения к программным модулям (и в последнюю очередь занимаясь тем, на каких серверах потом будут работать эти программные модули).

В Архимейте используются одинаковые принципы для описания деятельности людей, программ и оборудования. Во всех этих описаниях можно указать компонентное описание, модульное описание, размещение (проведите сами анализ того, как именно это делать — учтите, что в Архимейте терминология "компоненты", "модули" и "размещения" не используется! Ну, разве что есть объект, так и называемый location, "место"). Это лишний раз подчёркивает общность обращающихся к системному подходу инженерий: системной, программной, предпринятия. Используемые ими архитектурные языки при всей их разности оказываются довольно похожими по своей сути — в той мере, в которой они явно обращаются к системному подходу.

*Управление операциями*

Традиционно понимаемый менеджер-управленец — это пилот предпринятия, который следит за тем, куда лететь, нужно ли вообще лететь и когда стоит поменять курс. Стратегирование, клиентская область интересов. Тьма тьмущая книг "для пилотов", которые рассказывают о миссиях, целях, прогнозах, стратегиях, ценностях и политиках. Не меньше книг "для руководителей", которые рассказывают о том, почему организации — это не самолёты, и для них не нужно бортмеханика — ведь они сделаны из людей, а самолёт — из железа, и "уговаривать" железные части самолёта сотрудничать много проще, чем обеспечивать лидерство (leadership), катализирующее сотрудничество людей организации ради достижения поставленных пилотом-управленцем целей.

Операционный менеджер (организатор, инженер деятельности, применяющий практики инженерии предпринятия) это бортмеханик предпринятия, который следит за тем, чтобы самолет имел все необходимые для полета механизмы и летел. Чтобы организация могла работать, ее нужно организовать. Не очень много книг "для бортмехаников", в которых рассказывается о том, как распределить работы по людям, компьютерам и механизмам, и как поднять КПД по затрате ресурсов на единицу работы. Увы, в России для производства учат инженеров как конструкторов, и не готовят главных инженеров, как организаторов работ. Для госслужбы готовят людей, разбирающихся в государственной политике, а не организаторов-госуправленцев.

Авиатор начала века совмещал в себе все нынешние авиационные профессии — он был и пилотом, и бортмехаником, и стюардом в одном лице. Сейчас авиация развилась уже достаточно для того, чтобы не представлять себе пилота, копающегося перед вылетом в системе поджига топлива в авиатурбине. Трудно представить себе и бортмеханика, который бодро крутит фигуры высшего пилотажа, чтобы проверить сделанную им калибровку радара. А вот от современного менеджера ждут, что он с одинаковой легкостью а) поставит четкие цели и выработает стратегию их достижения [стратегирование] б) глаголом будет жечь сердца людей, чтобы они воспринимали эти цели как свои собственные [лидерство, “катализация сотрудничества”] и в) разработает и изготовит "производственный конвейер" [управление технологиями] и настроит его так, чтобы все работы выполнялись слаженно и успевались к срокам [операционное управление].

Но как бы не ожидали от менеджера универсальности, его плохо готовят к тому, чтобы он управлял не только людьми, но и операциями (operations — “функционирование”, работы. Управлять операциями — максимизировать проход целевых систем и услуг в сторону клиентов и денег со стороны клиентов).

Инженеры любят заявлять, что они берут на себя операционное управление — но у них просто не хватает времени и информации, чтобы посчитать критический путь или критическую цепь плана. Или наоборот, они продолжают называться инженерами (главным инженером, генеральным конструктором), но по факту управляют операциями — и тогда идёт провал в инженерии требований, своевременном выборе архитектурных решений, качестве системноинженерной проработки проекта. “Водитель, если ты одной рукой обнимаешь девушку, а другой рукой ведёшь автомобиль, то ты и то, и другое делаешь плохо”.

Это относится не только к России, но и ко всем развитым странам. Курс operation management много чаще входит в курсы подготовки инженеров, нежели

менеджеров (см. подробный обзор http://www.stedwards.edu/mba/draman/mgmt6305/pdf/Chapter1_p2.pdf). А ведь качественных миссии, видения и даже детальной стратегии оказывается недостаточно для эффективной работы, даже если удалось сделать так, чтобы команда их разделяла. Нужно ещё грамотно организовать операции — чтобы не срывались сроки окончания работ, чтобы вовремя и в нужном количестве закупались материалы, чтобы люди в организации могли быть уверены, что они выполняют самую необходимую в данный момент работу. Увы, менеджеров этому практически не учат: тамошний упор на стратегирование и лидерство, а управление операциями — это инженерия предпринятия, инженерная по сути дисциплина.

Основные дисциплины менеджмента (стратегирование, управление операциями, лидерство) объединяются в рамках популярных теорий менеджмента (в их приложениях для бизнеса называемых также бизнес-философиями — http://encyclopedia.thefreedictionary.com/Business+philosophies+and+popular+management+theories). К сожалению, большинство популярных теорий менеджмента относятся только к бизнесу, но не могут быть применимы без существенной адаптации к государственным, некоммерческим, церковным организациям ввиду разительной разницы в целях деятельности этих организаций и критериев измерения достижения этих целей. И только некоторые из них уделяют достаточно внимания организации операций (теория ограничений, TQM, JIT, lean management).

Проблема в том, что “просто менеджеры” обычно изучают операционное управление и связанные с ним менеджерские философии в крайне небольшом объеме. Вот, например, программы курсов по операционному менеджменту лучших 30 MBA-программ: http://web.archive.org/web/20070504104830/http://www.poms.org/EducationCourses/POMS_Syllabi.html. Из этих программ видно, что средний курс операционного менеджмента составляет примерно 20 классных часов.

Аналогичных курсов по организации операций для MPA (master of public administration) практически нет: трудно указать, как "государство преобразует входы (например труд, материал, знания, оборудование) в выходы (“товары” и “сервисы” — если их таковыми можно назвать в случае государства. “Сервис посадки в тюрьму”, “сервис привода в военкомат”, гм)". Вопрос о сходстве и различии организации операций в частном и государственном секторах поднимается (например, http://www.csus.edu/PPA/syllabi/undergraduate/spring2003/ppa140s.pdf, How Different are Private and Public Agencies Supposed to Be? Read: Paul Appleby, “Government is Different.” Graham Allison, “Public and Private Management: Are They Fundamentally Alike in All Unimportant Respects?”), но в большинстве учебных программ не отражен никак. Это только предстоит сделать. Тем не менее, движение за внедрение operations management по типу промышленной организации в public management уже идет. Вот некоторый набор ключевых слов: new public management, reinvent government, alternative service delivery и e-government. Главное слово в этом движении — performance (см., например, http://www.performanceweb.org/ — там есть и Lean Six Sigma For Government или Project Management for Government). "Государство как предприятие" на сегодняшний день воспринимается как факт, а не как один из возможных подходов.

В некоммерческих организациях также менеджеров почти не готовят к организации операций (см., например, четырехлетний курс по организационному управлению для некоммерческих организаций американской торговой палаты

http://institute.uschamber.com/required-core-curriculum/, где вопросы организации операций практически отсутствуют).

Ассоциация, которая занимается развитием дисциплины "управление производством и операционное управление" (Production and Operations management Society): http://www.poms.org/

Инженерия предпринятия и управление операциями

Инженерия предпринятия: архитектурное проектирование, "изготовление" и администрирование [управление операциями] эффективной работопроводящей сети из людей, компьютеров и механизмов (инженерный аспект: организация как "машина по достижению целей"). Каждый человек, компьютер, механизм в такой сети выполняет так или иначе регламентированные операции (работы, задачи, запросы, поручения), которые необходимо синхронизировать друг с другом и вовремя обеспечить материалами, чтобы их выполнение приводило к достижению глобального максимума для всей сети, а не для отдельных её узлов. Управление операциями — это про баланс объёма работ, ресурсов, качества и времени, а также способа, которым осуществляются работы (http://www.xprogramming.com/xpmag/kings_dinner.htm).

Организация при управлении операциями может быть представлена как "труба", по которой текут рабочие продукты — операционный менеджер интересуется возникающими очередями на их обработку (там, где очередь, там "запруда" в потоке, узкое место, ограничение — и там нужно внимание операционного менеджера для расшития этого узкого места). Эта метафора "очереди" является главной в ряде вариантов дисциплин операционного менеджера (так, в сообществе теории массового обслуживания, или по-русски operations research бытовала шутка: "Бомбардировщики подлетают к городу и становятся в очередь на обслуживание зенитной батареей").

Управление операциями как минимум в варианте operations research (по-русски это ещё переводят и как "исследование операций") у Берталанфи в его ранних работах по системному подходу классифицировалось как одно из приложений системного подхода к менеджменту — наряду с системной инженерией как приложением системного подхода к инженерии.

Управление операциями — это прежде всего интенсивная работа со знаниями, как и системная инженерия. Операции нужно сначала определить, а потом воплотить (или описать в плане обратной инженерии того, что получилось по факту, а затем определить к ним улучшения и реализовать их). Вот классификация знаниевых задач (Schreiber et al., 2000 по поводу CommonKADS, http://www.cs.vu.nl/~guus/papers/Speel01a.pdf):

Аналитические

— классификация

— оценка

— диагностика

— мониторинг

— прогнозирование

Синтетические

— дизайн (и конфигурационный дизайн)

— моделирование

— планирование (planning)

— определение сроков (sheduling)

— назначения (assignment)

Организация операций подразумевает преимущественно решение синтетических задач. Именно организация операций должна исправлять крен в сегодняшнем перекосе университетского менеджерского образования в аналитическую сторону (что отмечают многие критики программ MBA, например Генри Минцберг, который замечает, что выпускники программ MBA массово становятся не столько руководителями корпораций, сколько начальниками тамошних аналитических служб: они всё хорошо понимают, классифицируют, отчитываются — но не от них приходят решения по совершенствованию и развитию).

Пока же большинство курсов для руководителей содержат призывы проявлять больше творчества в этих важных вопросах управления операциями, но не дают конкретных знаний по дизайну, планированию, определению сроков и распределению ресурсов. Курсы операционного менеджмента, управления процессами, проектами, сетями поставок проходят обычно только "специалисты", а не руководители — при этом получается, что грамотно расставить и озадачить людей могут специалисты, не имеющие на это полномочий, а руководители имеют на это полномочия, но хорошо умеют только задавать общие цели на уровне стратегии и воодушевлять в порядке лидерства. Число специалистов в организации операций, не занимающих руководящие позиции, довольно велико. Организатор операций — это тот, кто пишет регламенты и составляет планы, и (обычно с ведома менеджера, хотя это прерогатива собственно руководителя!) объясняет людям, что им делать, с кем взаимодействовать, каких показателей достигать. К их числу можно отнести:

- management engineer и operation engineer. Особо распространены "инженеры управления" в здравоохранении (чтобы повышать пропускную способность госпиталя, организуя бизнес-процессы для бестолково суетящихся медсестер, врачей, техников медоборудования вокруг совершенно нетехнологизируемого и непрограммируемого процесса лечения).
- operations research analyst, как "в чистом виде" называют проектировщиков организуемых операций. http://www.collegegrad.com/careers/proft47.shtml — официальное описание, где можно убедиться, что различение между management и operations весьма условное (так, в этом официальном тексте говорится “Operations research” and “management science” are terms that are used interchangeably to describe the discipline of applying advanced analytical techniques to help make better decisions and to solve problems). В 1996 году в США было примерно 50000 таких работ, в 2002 году в США было примерно 61700 заполненных вакансий operations research analyst, в 2004г. 58000.
- management analyst (также называемый management consultant): http://www.ethics.state.pa.us/portal/server.pt/document/1280123/6_mgt_analyst_pdf указывает, что в США в 2004 году было 605000 таких работ, хотя и не все management analysts специализируются именно на организации операций.
- computer systems analyst, ибо значительное число специалистов по управлению ресурсами, проектному управлению, процессному управлению

работает сейчас в IT-службах. Это связано с тем, что в современной организации операций основным ее инструментом является компьютерная система, содержащая модель сети операций. В США в 2004г. было около 487000 системных аналитиков, значительная часть которых принимала участие в организации операций.

Управление операциями предполагает варианты специализации, известные под самыми разными именами:

- Операционный менеджмент (из Lingvo: operations management — управление операциями [производством]. Управление производственным процессом фирмы, в отличие от стратегического менеджмента, управления персоналом и других составных частей управления организацией; исторически первое название этой деятельности production management было изменено на operations management, т.к. по сути "производство" существует практически во всех организациях, и в том числе в сфере услуг, страховании, банковском деле и т. д., а слово production ассоциируется лишь с материальным производством). На английском языке общепринятое определение проще — Operations Management is the process by which an organization converts inputs (e.g. labor, material, knowledge, equipment) into outputs (goods and services). На русском языке наиболее часто используется до сих пор старая форма "управление производством" и много реже "управление операциями" или прямая калька "операционный менеджмент".
- управление цепочками поставок (supply chain management)
- управление проектами (project management) и управление процессами (process management), реинжиниринг процессов (process reengineering)
- планирование и управление производством (planning and production management)
- логистика (logistics)
- операционная стратегия (operation strategy)
- управление сервисными операциями (management of service operations)
- улучшение производительности (performance improvement)
- планирование ресурсов предприятия (enterprise resource planning) и управление ресурсами (resource management)
- get things done (GTD) — система персонального планирования

Все эти дисциплины разными словами обсуждают примерно одно и то же. Системный подход позволяет понять их похожесть и оценить различия, обусловленные развитием этих дисциплин для удовлетворения интересов самых разных стейкхолдеров в самых разных ситуациях и деятельностях.

Operations managmenet в самых разных вариантах переводится на русский одним и тем же словом — "управление", и нужно еще разработать русскоязычную терминологию в этой сфере. Вот, например, глоссарий ключевых терминов управления операциями в версии http://www.stedwards.edu/mba/draman/mgmt6305/pdf/Chapter1_p2.pdf: business environment, business processes, Business System Model, competitive edges, customer functions, decision-making functions, design, dimensions of a business, functional silo syndrome, goals, good, implementation, inputs, intangible service, inventory-intensive

businesses, management, manufacturing, measurements, operation, operations function, organization, output, policies, process, product, profit, resource management, schedule-intensive businesses, service, service industry, strategy, system, systems thinking, Throughput, transformation process, value-added. Видно, что это отнюдь не полный список. Вот много более полный список (но состоящий отнюдь не только из ключевых терминов): http://www.clamshellbeachpress.com/bookinfo.php?Id=1&Type=Learn+More — 408 страниц энциклопедии терминов операционного менеджмента. Вот ещё один глоссарий по supply chain и operations management — http://www.lindo.com/library/glossary.pdf (69 страниц).

Организация операций/операционное управление — это как организовать оптимальный по ресурсам максимальный (про)ход работ/денег через организацию. Пряность, работа и документы и деньги в английском обязаны течь (flow). По-русски это все "ходит", а не "течёт" — и поэтому к документам приделывают ноги, а не плавники, а деньги (хотя они и утекают) имеют хождение. С трубопроводной/потоковой метафорой, стоящей за словосочетанием "сетевая организация" (и даже за более распространённым "сетевым графиком"), в русском языке сложности — хорошую терминологию, эффективно задействующую метафорическое мышление, ещё только предстоит придумать.

Важно, чтобы через организацию проходило максимальное число работ, приближающих организацию к достижению её целей. Для этого нужно, чтобы работы в организации как минимум не мешали друг другу. Работы идут пошагово: операция за операцией. И возникают перекрёстные ритмы, как в джазе — когда различные цепочки работ с разным ритмом встречаются. Синхронизация всех этих ритмов, порождаемых разными исполнителями работ должна быть такой же, как в джазе: ноты нотами, но музыканты должны поглядывать друг на друга, подстраивая ритм и мелодику своего исполнения под то, что делают соседи. Музыканты слышат, что делают соседи. Работники могут и не подозревать, что происходит на соседних местах. Они даже могут не знать, какие им нужно играть ноты. Цели работ, планы работ и сам ход работ должны стать видимы: эту проблему решают организационная модель и система учета, компьютерное моделирование и расчёты планов.

Ещё одна сложность — время организовывания (build time) и время выполнения работ (run time) оказываются все более и более тесно переплетены ("рабочие места" перестали существовать десятилетиями в неизменном виде), поэтому архитектурная работа по определению предпринятия и операционное управление по его подстройке под текущую ситуацию оказываются тесно переплетены. И, конечно, кроме планирующей работы нужно ещё выдавать конкретные команды исполнителям, выбивающимся из плана — это уже не организация или управление, а руководство (буквально: “руками водство”).

Есть самые разные способы организовать управление операциями, но среди них можно выделить оркестровку (когда есть руководитель, который как дирижёр в оркестре, заставляет всех играть по одним и тем же нотам) и хореографию (когда никакого дирижёра нет: танцевальный ансамбль сам танцует, у него нет нот с указанием необходимого дальше движения и кто это движение выполняет, команда самокоординируется как партнёры в танце). Оркестровка и хореография как термины редко применяются в операционном управлении, но вот в управлении деловыми процессами (BPM, business process management) и workflow management как IT-дисциплинах это ведущие термины для описания того, как компьютеры делят

между собой работы/соорганизуют свои сервисы (http://www.bpminstitute.org/community/bpm-group/choreography-vs-orchestration).

Проектное управление

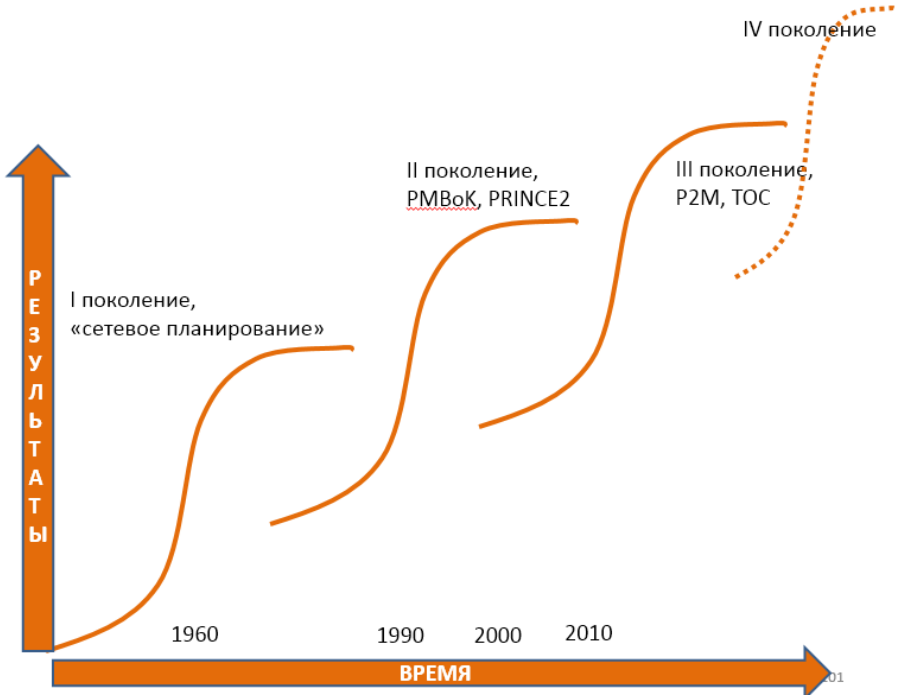
Проектное управление озабочено тем, чтобы спланировать и выполнить некоторую целенаправленную работу, ограниченную во времени и ресурсах.

Проект считается уникальным, имеет даты начала и окончания, связан с использованием ресурсов (людей, инструментов, материалов). Главная подальфа работ для проектного управления — график выполнения работ, проектное управление должно в какой-то мере гарантировать его составление и соблюдение (прохождение всех необходимых работ к моменту окончания проекта, с учётом ресурсных ограничений). Конечно, понятие проекта в разных школах проектного управления отличается, но все школы сходятся в том, что работы можно как-то предварительно (up front) спланировать, а затем выполнить план в том виде, как он задуман.

Разные школы проектного управления по-разному относятся к тому, как называть работы (например, PRINCE2 рекомендует названия работ заменять названиями основных рабочих продуктов, меняющихся в результате этих работ), как составлять график (в теории ограничений рекомендуют отводить на выполнение работы половину времени, заявленную экспертом по данной работе — но зато отдельно иметь "буфер проекта" для того, чтобы компенсировать неизбежные при этом задержки), как находит критические для контроля их выполнения работы (практики нахождения критического пути и критической цепи).

Оно существует в нескольких поколениях:

- первое поколение "сетевого планирования" (когда было предложено составлять "сетевые графики", в существенной мере облегчающие планирование заранее известных последовательностей работ — по этим графикам можно было найти "критический путь" (цепочку связанных предопределённой последовательностью работ, задержка каждой из которых приводит к задержке завершения всего проекта в целом).
- второго поколения (методологии PMI PMBoK, PRINCE2), в которой кроме самых разных аспектов планирования и контроля выполнения работ говорится также и о самых разных других аспектах управления проектом: стейкхолдерах и команде проекта,
- третьего поколения (методологии P2M/Project&Program Management, TOC/Theory of Constraints, LastPlanner/Lean Project management). Одним из ключевых положений третьего поколения этих методологий является рассмотрение всех проектов для данной совокупности ресурсов (т.е. проектов всего предпринятия в целом, а не проекта как отдельного предпринятия) в совокупности — т.е. переход к программам (совокупностям проектов) как основному объекту рассмотрения. Ибо нет другого способа управлять проектами, как перекидывать некритические ресурсы из одних проектов на критические задачи других проектов.
- Четвёртого поколения — исследования в области теории планирования, переходящие в задачи искусственного интеллекта и гибридным статистико-логическим вычислениям (вообще, теорию планирования относят к задачам искусственного интеллекта: пока алгоритма составления эффективного плана не придумано).




Традиционно проектное управление делят на управление портфелем проектов (если включить управление портфелем проектов и все проекты портфеля управляются тоже, то это управление программой — программа это множество проектов определённой темы, необязательно начинающиеся и заканчивающиеся одновременно), планирование проекта и контроль выполнения проекта. Но в третьем поколении проектного управления это деление не так уж очевидно.

Методы (наборы практик) проектного управления обладают некоторым “шовинизмом” (идеология превосходства с целью обоснования права на дискриминацию и угнетение других), т.е. пытается в рамках редукционистского подхода (в рамках представлений своей школы проектного управления) распространить свою дисциплину проектного управления на все смежные практики — от лидерства (ибо проектные команды нужно как-то формировать и воодушевлять) до инженерии требований (ибо для выполнения проектов нужны практики системной инженерии). Необходимо использовать из проектного управления лучшее, что оно может дать (т.е. оценку времени выполнения заранее запланированных работ и распределение ресурсов по работам такое, чтобы время выполнения этих работ было минимальным) и критически относиться к тому, что попадает на его “периферию” и много глубже изучено и лучше реализуется практиками других дисциплин — системной инженерии, лидерства.

### Управление процессами

Управление процессами имеет предположение, что заранее известные целенаправленные последовательности работ повторяемы — и это повторение относится не к самим работам (этот вопрос повторяемости видов работ решает представление о практиках, ситуационная инженерия методов описывает именно отдельные работы), а именно к последовательностям (цепочкам) работ.

Процессы часто называют business proccesses (по недоразумению переводятся как “бизнес-процессы”, хотя к бизнесу никакого отношения не имеют — business ведь это просто какое-то “дело”). Так же часто процессы путают с практиками (т.е. используют средства описания процессов для того, чтобы описать практики, получая неполные их описания — прежде всего описывающие “глаголы”, действия,

но опускающие описания альф и рабочих продуктов), наиболее часто эта путаница возникает при использовании набора стандартов ISO 9000. Так что в некоторых случаях правильно переводить process как "практика" (когда не говорится о последовательностях работ, а они только перечисляются). Так в ISO 15288 в английском оригинале говорится life cycle processes, но просто там в команде редакторов стандарта были люди-разработчики из стандарта ISO 9000 и поэтому там закрепилось слово "process". Конечно, в самом стандарте описаны практики: никакой последовательности выполнения работ, разворачивания работ во времени, упоминаний того, какие работы должны производиться в начале, а какие в конце, в ISO 15288 нет.

Что в проектном управлении называется "шаблоном проекта" и существует только в виде предоставляемых софтом проектного управления рабочих продуктов (но не альф, определяемых дисциплиной), то в управлении процессов — важнейшая альфа. И наоборот, что в проектном управлении важнейшая альфа — последовательность уникальных работ с конкретными временами исполнения, то в управлении процессов называется "экземпляр процесса", который также доступен главным образом в виде рабочего продукта процессного софта, но не обсуждается как альфа дисциплины.

Тем самым управление процессов хорошо применять тогда, когда нужно описать типовые последовательности работ, и для этого применяются даже специальные языки такого описания для workflow, когда работа идёт и через компьютеры и через людей — наиболее часто для этого сейчас используется OMG BPMN 2.0 (Business process model and notation, http://www.bpmn.org/).

Управление процессами трудно применять тогда, когда нужно дать оценку времени завершения отдельных экземпляров процессов ("проектов"), хотя при обилии экземпляров процессов и можно моделировать какие-то оценки загруженности ресурсов (людей и оборудования) и тем самым планировать проход потока работ по предпринятию-системе.

### Ведение дел/кейс-менеджмент

Ведение дел (case management и разные его современные варианты adaptive case management, dynamic case management, advanced case management — слово case тут того же происхождения, что и "судебное дело". "Управление делами" в русском закреплено больше за обслуживанием документооборота и/или материально-техническим обеспечением основной деятельности предприятий, поэтому мы и не используем этот вариант) — координационная и целе-ориентированная дисциплина, занимающаяся ведением дел от их открытия до закрытия, во взаимодействиях между людьми, вовлеченными в предмет дела и ведущим дело или командой дела (Case management. A coordinative and goal-oriented discipline, to handle cases from opening to closure, interactively between persons involved with the subject of the case and a case manager or case team).

Дело в русском языке (как и в английском) обычно отождествляется как с выполнением работ (деланием дела), так и с информацией по делу — case file (case folder) — уголовное дело, история болезни и т.д.

Ведение дел используется тогда, когда хочется отследить целенаправленную деятельность при невозможности заранее составить план (распределить работы во времени — т.е. использовать проектное управление) и последовательность (какая цепочка работ, что раньше из них, что позже — т.е. использовать процессное

управление) выполнения отдельных работ.

Дело — ситуация, обстоятельства или начинание, которые требуют набора действий для получения приемлемого результата или достижения цели. Дело фокусируется на предмете, над которым производятся действия (например, человек, судебное дело, страховой случай), и ведется постепенно появляющимися обстоятельствами дела (Case. A situation, set of circumstances or initiative that requires a set of actions to achieve an acceptable outcome or objective. A case focuses on a subject that is the focus of the actions such as a person, a lawsuit or an insurance claim, and is driven by the evolving circumstances of the subject).

Именно поэтому ведение дел относится к базированным на рабочих продуктах (и, соответственно, альфах) способам описания деятельности и близко к описаниям ситуационной инженерии методов, используемым в управлении жизненным циклом. Конечно, в ведении дел можно задать какой-то "шаблон" из последовательности действий (как и "шаблон проекта" в проектном управлении), но чаще речь идёт о задании каких-то правил выбора используемых следующими практик — и практики эти выбираются по правилам после оценивания ситуации после очередного такта работы (а то и вообще предлагаются новые практики). Это часто делается не "в софте", а при обсуждениях — судебных заседаниях, врачебных консилиумах, заседаний комитета по изменениям в инженерной компании.

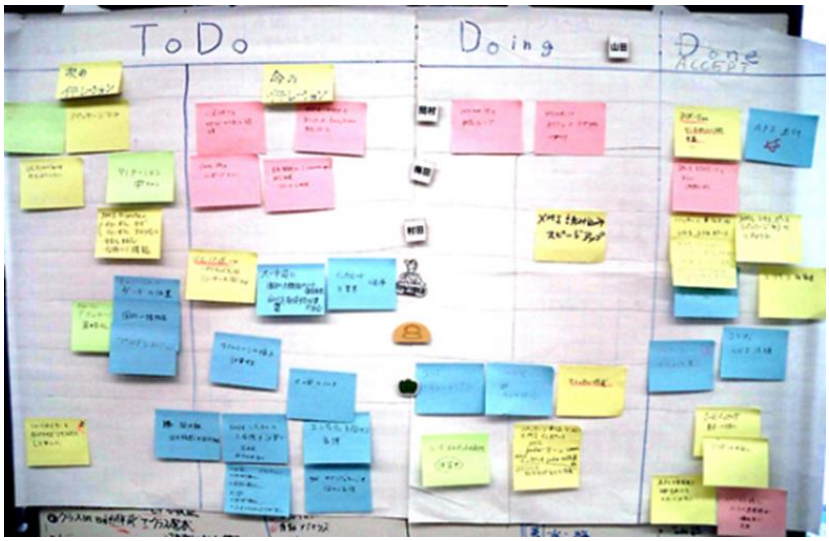
Ведение дел совместимо с ситуационной инженерией методов (ведение дел происходит главным образом в ходе работы, а ситуационная инженерия методов — перед выполнением работы). Практики в ситуационной инженерии методов описываются так, что выполнение любой операции связывается с другими операциями не непосредственно указанной последовательностью (жарим яичницу, затем проветриваем помещение), а пред-условиями и пост-условиями этих последовательностей ("когда хотим есть — жарим яичницу", "если в помещении дым, проветриваем помещение").

Если грубо, то управление делами занимается работами, которые носят в какой-то мере исследовательский характер (классические примеры — это судебные дела, в которых каждый новый свидетель и предъявленная улика вызывают неожиданные повороты дела и изменяют все планы, но в конце концов приводят к вынесению приговора; медицинские "кейсы", когда в больницу поступает пациент с неизвестным диагнозом, и каждый анализ и назначенное лечение приводят к неожиданным поворотам в планах врачей, но в конце концов больной выздоравливает и выписывается). К таким работам относится и инженерия требований, и архитектурное проектирование — когда каждое новое обнаруженное требование, каждый новый предложенный вариант архитектуры могут привести к переработке проекта-design (3D-моделей и расчётных моделей) и проекта-project (плана в смысле проектного управления), но в конце концов проектирование заканчивается и переходят к воплощению.

В программной инженерии используется issue tracking (иногда про это говорят как "младший брат для case management") и переводят сейчас часто как "управление задачами". Возникающие в программировании проблемы/вопросы/дела (issues) часто относятся к непредвиденным в планах и неожиданным для процессов, непредусмотренным практиками ошибкам, поэтому их нельзя запланировать в рамках проектного управления, нельзя предусмотреть и учесть в рамках процессной работы, но они всё одно требуют учёта. Поэтому в программной инженерии используют специальный класс программного обеспечения — issue trackers (системы управления задачами).

Практически во всех современных инженерных софтах управления жизненным циклом (PLM-системах, product life-cycle managmenet) находятся именно issue trackers, называемые обычно “системами управления изменениями”. Каждое issue/задача/проблема/вопрос/изменение понимается как запрос на изменение (Engineering Change Request). После обсуждения того, что и кто должен изменять, этот запрос на изменение превращается в поручение (Engineering Change Order), а после выполнения работы — извещение о том, что запрос удовлетворён, начальная проблема закрыта (Engineering Change Notice).

Вот простейший вид issue tracker в виде стикеров на ватмане:




Обратите внимание, как команда проекта меняла состояния, через которые проходит issue: сначала состояние было названо Done, но потом команда поняла, что “сделать” это ещё не “принять сделанное” (помним про трансакции DEMO) — и переименовала колонку.

Не слишком похоже на диаграммы Гантта и прочие рабочие продукты управления проектом? Да, это совсем другое view на работы.

### Управление проектами и управление жизненным циклом

Инженерную дисциплину управления жизненным циклом регулярно путают с менеджерской дисциплиной управления проектами. Это распространяется на попытки использования ситуационной инженерии методов (используемой в ходе создания предпринятия перед выполнением самих производственых работ) и сопутствующего ей ведения дел (в ходе выполнения производственных работ) в качестве эдакого дубля для проектного управления. Правда в том, чтобы использовать оба подхода, а не просто замещать проектным управлением инженерную работу по управлению жизненным циклом.

Управление жизненным циклом — это про использование основанных на системноинженерных дисциплинах (инженерии требований, инженерии системной архитектуры и т.д., формулируются в терминах альф) практик решения инженерных задач, а проектное управление — это про *контроль оформления* результатов этого решения в конкретных рабочих продуктах.

Оценки разработчиков OMG Essence таковы, что одна альфа свидетельствуется часто десятком рабочих продуктов — да ещё эти продукты часто разные для разных состояний альфы. Рассуждения в ходе разработки ведутся содержательные, концептуальные, инженерные, "в терминах альф". Рабочие продукты только

оформляют результаты этих содержательных рассуждений, рабочие продукты — это форма для фиксация содержания (при всём уважении к неразрывности формы и содержания).

Инженеры говорят содержательно, в терминах альф: "архитектура готова", "пользователи удовлетворены". Менеджерам всё равно, что там с содержанием, поэтому они предпочитают говорить в терминах рабочих продуктов: их наличие или отсутствие легко проверить, а факт содержательности тоже легко проверить путём получения подписи на них какого-то эксперта ("архитектурное эссе согласовано, но ещё не подписано", "подписи заказчика на акте приёмки-сдачи уже получены"). Менеджерам трудно проверить "архитектура готова", но легко проверить "файл с принципиальной схемой опубликован". С точки зрения инженеров три месяца идёт работа над архитектурой, а потом три дня она оформляется в виде рабочих продуктов. С точки зрения менеджеров три месяца ничего не происходит, а потом бац — "архитектурные рабочие продукты готовы".

Менеджеры предпочитают структуру проекта-design в системах управления конфигурацией (PLM-системах) делать "по томам документации", это облегчает контроль готовности и передачу заказчику всех запланированных рабочих продуктов. Инженеры предпочитают хранить "по сборкам" (т.е. не в соответствии с разбиением по документации, а в соответствии со структурой системы — компонентной и модульной декомпозиции), что не помогает для контроля готовности и передачи заказчику, но хорошо помогает при собственно разработке.

Управление жизненным циклом признаёт тесную взаимосвязанность и параллельное (одновременное) практикование самых разных дисциплин в ходе инженерного проекта. Контрольные вопросы — это вопросы прежде всего к самому важному, т.е. очень небольшому числу аспектов инженерного проекта. Контрольные вопросы ни в коем случае не являются полными для контроля проекта! Они просто напоминалки, попытки обратить внимание на отдельные важнейшие моменты! Они не заменяют собой детальных планов, в которых учтены все требуемые рабочие продукты, не заменяют собой все наборы требований, каждое из которых требует проверки, не заменяют собой всех других вопросов, которые задаст в те или иные моменты работы команда инженерного проекта.

Ответы на контрольные вопросы приводят к формулированию дел (issues/tasks/cases) — постановке проблем, заданию задач, задавания вопросами. Эти отдельные дела обычно не учитываются ни в одном из планов, но тем не менее, их нужно выполнять. Все дела формулируются по возможности в содержательной форме (альф), а не в форме оформления рабочих продуктов (хотя и могут быть отдельные дела, связанные именно с недостатками оформления).

У "управленцев проектами" в голове план выпуска рабочих продуктов, как проходящих по "трубе предпринятия" (метафора потока, хода работ) — то, что легко проконтролировать по форме при прохождении работ и передаче их от одного исполнителя к другому. И вот во что превращается Essence с его контрольными вопросами при попытках его изменить с моделью "проектного управления" в голове:

- по стандарту Essence детализация на подальфы требуется только там, где "команда не понимает" или "команде трудно дать однозначный ответ". Но после перехода к бюрократическому контролю детализация на подальфы потребуется везде, просто как ещё одно средство детализации контроля. Из механизма дебюрократизации (уменьшение объема необходимых рабочих

продуктов) получается ровно обратное: попытка бюрократизировать творческий инженерный процесс. Лишние подальфы часто вызывают необходимость подготовки новых рабочих продуктов, по-разному повторяющих одну и ту же информацию. Для некоторых проектов лишнее документирование может быть хорошо, но для большинства проектов это непроизводительная трата ресурсов, отход от принципов lean (не выполнения ненужной работы).

- поскольку содержательные ответы команды бессмысленны и плохо проверяемы (а часто инженеры нагло и необоснованно врут друг другу и менеджерам, а менеджеры тоже врут друг другу и инженерам, когда дают положительные ответы на контрольные вопросы и закрывают глаза как раз на те риски, которые призваны вскрыть контрольные вопросы), то делается попытка немедленно привязать альфы к рабочим продуктам и вопросы задавать уже про рабочие продукты. Это недопустимо! Разговор про "требования" — это не разговор про "техническое задание" или "опросный лист", а разговор про "архитектуру" — это не разговор про "четвёртый лист схемы автоматизации"! В итоге получаем тьму довольно бессодержательных вопросов про готовность рабочих продуктов к сдаче, а не вопросов про проработанность решений для перехода к следующим состояниям альф.
- набор карточек для вопросов к рабочим продуктам, развёрнутый по линии времени становится просто планом из PRINCE2 (названия задач представляют собой названия готовых рабочих продуктов), WBS верхнего уровня которого наследует рубрикацию рабочих продуктов, оставленную в наследство от альф Essence.

Проектное управление и ведение дел: не "или", а "и".

По факту в проекте одновременно используются самые разные определения деятельности и поддерживающий их софт:

Перед выполнением проекта (в ходе инженерии предпринятия):

- Определения практик (в виде регламентов, при определении формы жизненного цикла, методик проектирования, ГОСТов и т.д.)
- Процессы (служба качества)

Изредка — шаблоны проектов  В ходе работы предпринятия:

- Задачи в переписке (как в электронной почте, так и в официальной переписке). Часто говорят о "поручениях".
- Дела в протоколах совещаний и других бумажных документах
- дела в issue trackers, в том числе выявленные при работе с карточками Essence
- Планы проектов (то, что удалось спланировать up front) в системе проектного управления.

Конечно, это далеко не полный список.

Это риторический вопрос, будут ли все эти списки дел/работ согласованы между собой по названиям, датам, ресурсам, ответственности и т.д. — и какие из этих явных (в случае проектного управления) или неявных (в виде списков дел в issue tracker) планов будут актуализовываться и мониториться в выполнении. Очевидно,

что требуются огромные усилия по пониманию того, как устроено планирование дел в проекте — как тех дел, что можно запланировать перед началом актуальной работы, так и тех дел, которые заведомо не попадут в такие планы (типа исправления конфигурационных коллизий, проявляющихся только в ходе работы).

Конечно, нужны и управление жизненным циклом с кейс-менеджментом "по содержанию", и управление проектом с план-графиком и выдачей рабочих продуктов "по форме". Ибо по issue трудно рассчитать критическую цепь (но можно обсуждать проблемы), по рабочим продуктам трудно обсуждать продвижение в решении содержательных задач (но можно обсуждать оформление решений). Мамы всякие нужны, мамы всякие важны: разные дисциплины отвечают на разные вопросы.

По какому плану будет вестись проект инженерами? Конечно, будет работать ведение дел (issue tracking) в содержательных терминах (альфы), а не управление проектами: крупные пункты плана проектами ("директивный график" — задаваемый соглашением со стейкхолдерами "политически" и основанный на экспертных оценках, игнорирующих потом выявляемые проблемы) представляются для управления делами крупными целями, но более мелкие задачи формулируются "с голоса" по ходу проекта и попадают в самые разные распорядительные документы и даже проходят мимо документов — поручения "в рабочем порядке", пункты протоколов совещаний, пункты отдельных приказов, и issue в каком-нибудь issue tracker. Конечно, инженер предпринятия должен по возможности минимизировать число систем, в которых ведётся учёт дел (а часто и минимизировать число планов проекта: в крупных проектах легко найти пять разных планов проекта, не совпадающих друг с другом!).

Но в момент прохождения гейтов (принятия решений "Go — No Go" между стадиями жизненного цикла) все эти планы проверяются на соответствие — понимание рисков инженерами и менеджерами согласовываются. Это хорошо понимают разработчики систем ведения дел ("управления задачами"). В этих системах стремительно появляются возможности систем проектного управления (например, показать диаграмму Гантта для имеющихся в системе задач).

### Управление мероприятиями

Управление мероприятиями (Event management is the co-ordination, running and planning of all the people, teams and features that come together to create every kind of event, http://www.eventbusinessacademy.com/why-events/what-is-event-management, часто переводят как "событийный менеджмент") предназначено главным образом для управления проведением какими-то собраниями людей: Олимпийских игр, концертов, конференций, фестивалей. Тут мы приводим в пример "управление мероприятиями" просто для того, чтобы показать огромное разнообразие деятельностей по производству разного типа целевых систем и сервисов, которое требует создания самых разных видов обеспечивающих (enabling) предпринятий.

Конечно, можно использовать для проведения концерта знания по системной инженерии, но более правильно было бы использовать знания профессионального управленца мероприятиями. Но если вам придётся заниматься самыми разными деятельностями, то лучше бы владеть знаниями по инженерии предпринятий в целом — и использовать системный подход, чтобы иметь возможность собрать все эти отдельные знания в одно целостное представление о целевой системе или сервисе, а также о создающем его предпринятии.

Специализированное знание в конкретном деле всегда выигрывает, применяйте более общее знание только тогда, когда вам не хватает специализированного знания. Но нет ничего полезней, чем более общее знание (а особенно системный подход), когда вам нужно быстро разобраться со специализированным знанием: оно будет представляться вам не таким уж и специализированным, вы освоите его много быстрее и сможете перенести на него опыт и других известных вам деятельностей.

Финансы

Важная часть инженерии предпринятия в целом и управления операциями в частности — это финансы, в которых нужно разделить управленческие финансы (использование для принятия операционных и инженерных решений) и бухгалтерию (”посмертный учёт”, главным образом в целях налогового учёта). Управленческие финансы имеют множество разных школ, определяющих разные способы учёта и планирования денежных потоков (помним, что управление операциями — это прежде всего логистика, “потоки и запруды на них”, “операции и очереди к ним”, “рабочие станции и входные буферы рабочих продуктов”).

Каждая школа задаёт свои альфы, и нужно думать, какими практиками на основании каких дисциплин пользоваться при создании предпринятия.

Нужно, например, запомнить, что “себестоимость” — это альфа, которая не признаётся некоторыми школами управленческого учёта (например, в throughput accounting, http://en.wikipedia.org/wiki/Throughput_accounting) в силу того, что там смешиваются переменные и постоянные издержки, которые в учёте нельзя смешивать. Годовое планирование бюжета не признаётся в школах beyond budgeting (http://www.bbrt.org/beyond-budgeting/bb-mental.html — упражнение: сравните тамошние принципы с теми, которые изложены в разделе “инженерия организации” и найдите в тексте раздела соответствующие тамошним принципам оговорки), в которых предлагается иметь не годичное окно планирования (”годовой бюджет”, утверждаемый раз в год), а гораздо более гибкое трехмесячное или даже месячное скользящее окно принятия решений о направлениях расходов.

## *Управление знаниями, НСИ, (справочными и мастер, а также проектными) данными*

Инженерия и предпринятия-киборги.

Если речь идёт о материальных рабочих продуктах (а хоть и о бумажных документах, или даже неструктурированных электронных документах), то никаких проблем для описания работы инженерии и операций нет. Но ситуация резко меняется, когда у нас в предпринятии появляется большой объем работы с:

- информацией (фактами, сведениями, приказами, требованиями, мнениями). Обсуждение информации затрагивает содержательные вопросы – значение информации в контексте ее использования («смысл»). Что эта информация означает для нашего инженерного проекта, что из этой информации следует? В чем смысл именно такой информации, а не другой? Какой ситуации в реальном мире соответствует эта информация? Информация в рабочих продуктах позволяет судить о состоянии альф, проводить содержательное обсуждение. Содержательное рассмотрение обычно никак не связано с обсуждением способа записи, соглашениями об именах, синтаксисом, особенностями представления информации на носителе. Это именно содержание, а не форма. Информация о том, что 2*2=4 обсуждается

содержательно (что это именно 4, не 3 и не 7), а не с точки зрения её представления (например, использованной системы счисления для записи цифр), или на каком носителе она записана. Инженеры работают с информацией.

- данными (представлением информации с использованием какого-то формализма. Например, при обсуждении данных 2*2=4 можно обсуждать формализм – арабские цифры или римские). Данные абстрактны, т.е. не существуют в материальном мире. Действительно, одни и те же данные (например, текстовая строка «труба» в кодировке UNICODE или даже большая база данных со сложной схемой) могут быть представлены на самых разных экземплярах носителя, в том числе и резервных (бэкапы). На всех этих экземплярах носителей данные остаются одними и теми же, и обсуждаются именно как данные. С данными работают программисты и прочие айтишники.
- информационными рабочими продуктами (иногда говорят "информационными объектами"), т.е. чертежами, отчётами (бумажными и даже электронными – файлы), хранилищами данных на компьютерах (предназначенные для хранения данных в структурированном виде, без повторений и с возможностью однозначного нахождения нужных данных) и любые другие физические объекты, содержащие данные. На разных информационных объектах/носителях информации (например, в газете и в хранилище данных) могут находиться одни и те же данные (например, текстовая строка «труба»). Рабочие продукты можно пощупать, они находятся в физическом мире, они не абстрактны.

Данные не живут без носителя. Так, «файл» – это обычно маленькие магнитные частички, специальным образом ориентированные, расположенные во множестве разных мест диска из немагнитного материала. Данные из файла могут затем попасть в оперативную память – и стать там уже не магнитными частичками с разной ориентацией в пространстве, а участками полупроводниковой микросхемы с разными напряжениями.

Современное предприятие представляет из себя киборга, в котором коммуникация (логистика информации) между людьми опосредуется информационными системами, а рабочие продукты для отражения состояния альф стратегирования, инженерии, предпринятия во всё большей степени представляют собой информационные рабочие продукты, логистика которых означает пересылку данных с одного носителя на другой. Вместо слесарей и обкатчиков клюквы (http://www.aup.ru/docs/etks/etks-51/144.htm) у нас появляется огромное число работников, для которых очень трудно представить Toyota production system с её тележками и минимизацией входящих очередей болтов и клюквы к отдельным рабочим местам. Операции для информационных работников трудно описать не только потому, что трудно описывать работы "думания" и "творческой коммуникации", но и потому как в эти практики входят паттерны нечеловеческой работы — паттерны "думания" и "коммуникации" компьютеров, которые оперируют данными (т.е. представлением информации с использованием какого-то не слишком связанного с содержанием этих данных более-менее универсального формализма).

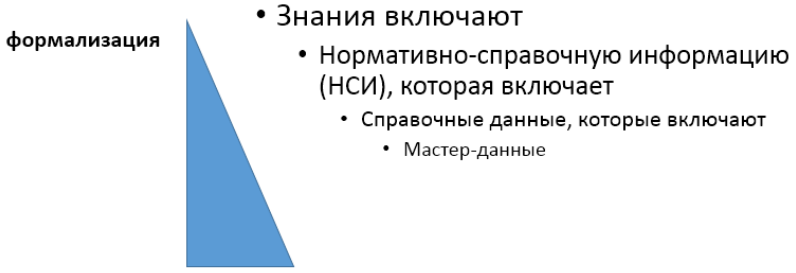
Но это не означает, что мы не можем попытаться попробовать говорить об операциях киборгов (у которых есть кортекс и экзокортекс) так же, как мы говорим об операциях "просто людей". В головах (кортексах) и экзокортексах (личных и корпоративных компьютерах, забудем о далёком прошлом со шкафами бумажных

документов, а также бумажных записных книжках) людей идут какие-то инженерные работы с информацией/данными, и информация/данные передаются между ними и хранится где-то между обработками. Важно ведь не просто сработать с информацией, чтобы извлечь из неё что-то интересное силами одного исполнителя-гения (будь то человек или компьютер), но и научиться выстраивать информационный конвейер предпринятия-киборга, самого состоящего из имеющих разные компетенции и по-разному загруженных работой киборгов (людей и их инструментов-компьютеров) так же, как сначала Форд, а затем люди в Toyota научились выстраивать свой автомобильный конвейер из людей и механических инструментов. И в этом информационном конвейере предпринятия какие-то данные могут быть порождены пару лет, а какие-то данные даже пару сотен лет назад, и попадут на этот конвейер не только из голов работников-людей через клавиатуру или распознаванием речи, но и в виде справочных данных из уже давно существующих текстов, баз данных и т.д.

Инженерия знаний и управление знаниями.

Мы будем пытаться единообразно говорить об организации как работы с информацией/данными, так работы с физическими объектами. Сегодня мы рассмотрим дисциплины/практики операционной работы с информацией/данными, которая используется во множестве проектов/предпринятий, на нескольких стадиях жизненного цикла, и/или интересует множество выполнителей. Информация и данные, которая используется во множестве проектов/предпринятий называется:

- знаниями (если акцент делается на кортекс и том, что "внутри головы", а также обсуждения потребностей в этом знании. О формате представления умолчим, ибо употребляющие слово "знания" избегают разговора про форматы и формализмы).
- нормативно-справочной информацией (если акцент делается на эксплицитно представленном знании в любой их форме — неструктурированной/полнотекстовой, равно как структурированной в виде объектных, реляционных, графовых баз данных, файлов данных компьютерного моделирования и т.д.)
- справочными данными (если акцент делается на структурированной части НСИ, представленной единообразно, а рассмотрение не столько содержательное-инженерное, сколько менеджерское-логистическое — то есть “хранение и доставка данных по назначению”, абстрагированное от смысла данных)
- мастер-данными (если акцент делается на справочных данных, подвластных единому центру администрирования — обычно в рамках одного предприятия-юрлица)

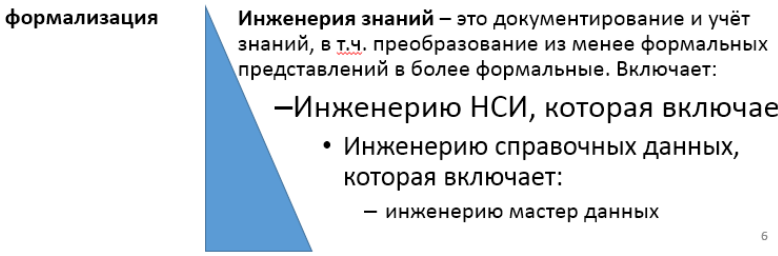



С этими объектами работы идёт как инженерная/информационная работа (преобразование в соответствии с технологией получения целевой системы), так и

операционная ("управление" в его учётном/конфигурационном и логистическом/маршрутизационном смысле, акцент на формирование данных в отвязке от их содержания, их учёт и их передачу между местами содержательной обработки).

Инженерии тут аж две:

- инженерия знаний (в части формы представления): инженерия знаний/НСИ/справочных данных/мастер-данных — это преобразование из менее формальных представлений в более формальные. "Инженерия знаний" традиционно относится к трансформации человечьего знания в нечеловечье/компьютерное (http://en.wikipedia.org/wiki/Knowledge_engineering). То есть это выковыривание знаний (как explicit, так и tacit) из человечьей головы (кортекса), текстов на естественном языке (находящихся в "экзокортексе") и кодирование его в формальные структуры, понятные компьютерным программам. Это программирование/моделирование/онтологизирование, которые всё суть одно.




  Инженерия знаний при этом включает инженерию НСИ, инженерия НСИ включает инженерию справочных данных, а инженерия справочных данных включает инженерию мастер данных. Чем менее формально представление, тем больше обсуждение информационной составляющей, а чем более формально представление, тем больше обсуждение составляющей данных. Кроме того помним, что (как и в любой инженерии) всегда принимается решение “делать самому или купить готовое” — огромное количество справочной информации уже существует, и нужно тщательно думать, создавать ли очередно классификатор, модель данных или стандарт, или же повторно использовать уже имеющиеся.

- дисциплинарная, т.е. целевая для деятельности предприятия инженерия (в части информации о целевой системе предпринятия, содержания/предмета работы выполнителей-инженеров): системная инженерия и инженерия по специальности, в том числе порождающее проектирование (generative design: когда “думает” компьютер, а не инженер — т.е. компьютер выступает в роли большей, чем “редактор”) и прочие содержательные обработки знаний/справочных данных, вплоть до "думания" при помощи инженерных программ искусственного интеллекта. Эта инженерная содержательная часть работы со знанием остаётся вне предмета нашего рассмотрения. Когда говорят об "инженерии знаний", вовсе не имеют в виду какую-то конкретную последующую работу с этими знаниями (придумывание новых знаний при помощи компьютера, использование текущих знаний, использование "настоящего искусственного интеллекта" и т.д.), речь идёт главным образом о формировании адекватного для использования в этой последующей работе представления знаний в каком-то компьютерном формализме.

Традиционно "управление знаниями" относится к операциям — как и любое

"управление". Оно включает управление конфигурацией и управление изменениями (учёт), а также логистику (поиск и доставку по потребности) знаний, находящихся в (http://ailev.livejournal.com/631926.html, 2008):

- человечьей голове (байки про важность tacit knowledge и незаменимость человеческой интуиции). Вотчина психологов, управленцев персоналом, когнитологов (когда нужно выковырять это знание, сделав его explicit).
- текстах на естественном языке (нормах и стандартах, переписке, учебниках — все эти "просто поиски", "семантические поиски", "автоматическое аннотирование" и пр.). Вотчина компьютерных лингвистов, программистов и администраторов "систем управления контентом" (ах, опять это "управление" — учёт и логистика "контента", полностью абстрагированного от его содержания!).
- нечеловечьих/компьютерных структурах данных, подчиняющихся различным формализмам (базах данных, инженерных моделях и т.д. — федерирование и интеграция данных, корпоративные шины предприятий, сервис-ориентированные архитектуры и пр.). Вотчина айтишников-системщиков, которые от железа уже ушли, а к инженерно-дисциплинарной работе с информационным содержанием данных ещё не пришли.

"Управление знаниями" сегодня используют так же бездумно, как "управление требованиями": большинство менеджеров считают, что требования появляются не в результате инженерии требований (инженерной дисциплины), а в результате управления требованиями (операционной дисциплины: учёт/управление конфигурацией и изменениями требований и логистика/маршрутизация/доставка требований и их полуфабрикатов туда и тогда, где и когда они нужны). Ну да, булки растут даже не на деревьях, а прямо в машинах по развозке хлеба. Производителем сообщения (инженером) является принесший его гонец (операционный работник). Увы, жизнь устроена сложней: для знаний нужны и инженеры-придумщики-изготовители, и операционисты-менеджеры с их учётом и логистикой.

Как и в случае инженерии знаний, управление знаниями включает в себя управление НСИ, управление НСИ включает в себя управление справочными данными, управление справочными данными включает управление мастер-данными. Другое дело, что дисциплина (набор практик) "управление знаниями" включает в себя также и практику управления доступом одних людей к tacit knowledge других людей, хотя мало кому удалось понять, в чём же именно заключается такая практика.

Эту "матрёшку" будет довольно трудно различать в живой речи на производстве — люди очень любят метонимию в варианте синекдохи: называя целое именем части, да и наоборот тоже. Ну, и никаких "операций со знаниями", "операционного менеджмента знаний", а только "управление знаниями". Даже "операции с данными", скорее всего, будут означать инженерные преобразования данных, так что не ждите какой-то консистентности использования предлагаемой терминологии. Ничего, мы будем к этому толерантны, а наши лингвистические программы и онтологические к ним настройки будут это учитывать.

Инженерия знаний — это порождение и документирование/постановка на учёт тех объектов знаний (данных!), которые потом будут путешествовать по логистической сети между выполнителями-"думателями"-компьютерами. Поэтому её можно включать в управление знаниями примерно на тех же основаниях, на каких управление конфигурацией включают в инженерный менеджмент/управление

операциями, и считать "менеджерской дисциплиной" (не зависящей от предметной области, к которой относятся знания по их содержанию).

С другой стороны, инженерия знаний много сложней, чем управление конфигурацией и управление изменениями материальных компонент целевой системы. Это не просто нарезка знания на содержательные куски, удобные для коммуникации, и учёт этих знаниевых конфигурационных единиц и их изменений, как материальную систему нарезают на конфигурационные единицы и затем ведут их учёт. Нет, там много особенностей по тому, как "выковыривать" знание из одних форматов и кодировать в других без потери информации/содержания: в материальном мире обычно такие задачи не требуют специальных методов, разобрать часы на детальки может и маленький дитёнка (в отличие от разборки смутных представлений специалиста о его способах работы и представление их в виде текста, или разборки пяти мутных текстов в семантическую сетку). Есть и иная традиция, в которой управление знаниями затягивают внутрь инженерии знаний (это часто: управление требованиями затягивают внутрь инженерии требований — а ведь управление требованиями это управление знаниями в части требований, инженерия требований — инженерия знаний в части требований, хотя и приправленная содержательной инженерией для формулирования содержания требований).

Мы не будем в этом разбираться, а просто припишем инженерию знаний (и все эти "инженерии НСИ", "инженерии справочных данных", "инженерии мастер-данных") к операционной деятельности, рядом с управлением конфигурацией и управлением изменениями — это всё очень рядом с PLM (хотя поставщики PLM-систем этого и не осознают, и не дают такого инструментария. Ну, разве что Dassault Systemes прикупила Exalead, но это всё-таки не совсем то). Вот такой вот операционный шовинизм по отношению к инженерии знаний, ибо у этих "инженеров по знаниям" основное — это "знание о знании", от них не требуется инженерного "знания о целевой системе и способам её получения", знания о железе и софте, химии и прочностных расчётах. А "настоящие инженеры" (системные и инженеры по специальностям) у нас в предпринятии будут заниматься не формализацией-деформализацией знаний/данных как таковых, а содержательной инженерной работой с информацией по созданию целевой системы — добычей и использованием справочной информации, решением традиционных инженерных задач. И будут поэтому формализовывать/деформализовывать знания только по сопричастности к их содержательной инженерной деятельности.

Помним, что все приведённые разведения (прежде всего инженерии и операций) очень условны. Так, Ассоциация инженерного менеджмента (http://www.asem.org/) предпочитает говорить о непрерывном спектре инженерных и менеджерских дисциплин (где "чистая инженерия про железо" и "чистый менеджмент про людей" находятся на краях, а "инженерный менеджмент" где-то посерединке, и включает в себя как раз главным образом "операционный менеджмент" в самых разных его вариантах — проектное управление, управление цепочками поставок и т.д.). Условность отнесения рассматриваемых вопросов инженерии и управления знаниями, равно как сближения управления конфигурацией и данными (помним про одноимённую ассоциацию — http://www.acdm.org/), и замешивания их в одну кучу в части "операционного управления" должна быть предельно ясна.

* * *