6.7. Управление программным проектом

«Такую личную неприязнь я испытываю к потерпевшему, что кушать не могу…»
из к/ф «Мимино»

Меньше всего хотелось бы упоминать в управлении программными проектами о руководстве и эффективном менеджменте, помятуя известную народную мудрость: «Не умеешь работать, иди руководить». Поэтому возьмем за основу другой термин, который имеет идентичный перевод в данном контексте: management и control переводятся как управление, только первый подразумевает больше организационные меры, а второй акцентируется на технологическом аспекте. Короче, будем рассматривать проблему с позиций *ситуация под контролем*.

Своды знаний и стандарты

Начать придется с менеджмента. Управление программным проектом – это проектная деятельность, а управление проектами - **PM** (**Project Management**) – стандартизованная категория менеджмента. В ней, аналогично программной инженерии, существует свой **«Свод знаний по управлению проектами» PMBOK (Project Management Body of Knowledge)** [6-9], охватывающий следующие перечень деятельностей:

- управление интеграцией проекта (project integration management);
- управление содержанием проекта (project scope management);
- управление сроками проекта (project time management);
- управление стоимостью проекта (project cost management);
- управление качеством проекта (project quality management);
- управление человеческими ресурсами (project human resource management);
- управление коммуникациями проекта (project communication management);
- управление рисками проекта (project risk management);
- управление поставками проекта (project procurement management).

Баня, музыка, кино и мост. Проектная деятельность в разных областях имеет свою специфику. Для программной инженерии наиболее удачные аналогии и метафоры можно найти там, где результат имеет нематериальный характер, например, в кинопроизводстве или постановке спектакля.

В программной инженерии вопросы управления программными проектами отражены в отдельном документе SWEBOK «Управление программной инженерией» (Software Engineering Management). Основные разделы идентичны стандарту IEEE (ISO/IEC) 12207 в части «Процесс управления» (Management Process):

- инициирование и определение содержания;
- планирование программного проекта;
- выполнение программного проекта;
- обзор и оценка;
- закрытие проекта;
- измерения в программной инженерии.

Кроме того, отдельные вопросы управления проектом обсуждаются в других документах SWEBOK:

- требования к программному обеспечению (Software Requirements);

- конфигурационное управление (Software Configuration Management);
- процесс программной инженерии (Software Engineering Process);
- качество ПО (Software Quality).

Специфика управления программным проектом

Если же отвлечься от сухих стандартов, то для поддержания ситуации под контролем в любом проекте необходимо:

- организация команды исполнителей;
- планирование работ;
- оценка сроков исполнения, стоимости, объема работ для будущего периода, учет из для прошедшего периода – метрика проекта;
- оценка потенциальных угроз проекту – рисков.

Этим, собственно, и занимается **Project Management.** В программной инженерии необходимы поправки поправки на специфику отрасли:

- специфические риски, связанные с качеством персонала, оценками трудозатрат, адекватностью представлений и требований;
- различные принципы организации и самоорганизации исполнителей, широкий спектр методологий, в каждой из которых имеется собственная структура менеджмента;
- особенности метрик программного продукта, метрик качества кода;
- особенности и структура жизненного цикла ПО.

Особенности программной инженерии как проектной деятельности были отмечены в п.1.2.. Нет нужды повторять, что это все имеет прямое отношение к управлению проектом. Существует также связь методологии разработки с основными метрическими характеристиками проекта, учитываемыми в управлении [6-13] (рис.6-41).

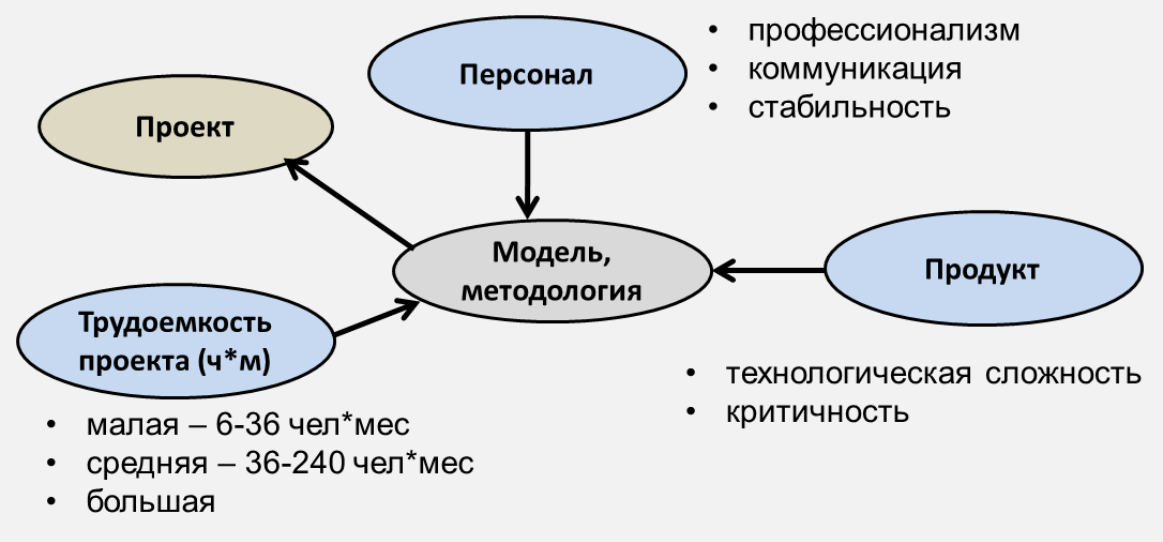

Рис.6-41. Факторы, влияющие на выбор методологии разработки проекта.

Под *тяжестью* методологии понимается количество элементов управления - артефактов, деятельностей, под *плотностью* - степень их детализации и связности с другими. Выделены следующие зависимости:

- чем больше команда, тем тяжелее используемая методология;
- плотность методологии пропорциональна критичности проекта;
- чем тяжелее методология, тем выше стоимость проекта;

- самая эффективная форма коммуникации - непосредственное общение.

Методология и организационная структура компании

Программный проект и коллектив исполнителей существует в рамках компании, в организационную структуру которой он вписывается. Традиционно наиболее распространенной формой является **функциональная структура** - подразделения, ориентированная на функции или на исполняемые виды работ - руководство, финансовое планирование, ресурсное обеспечение, снабжение, инфраструктура, web-разработка, клиентские приложения (рис.6-42).

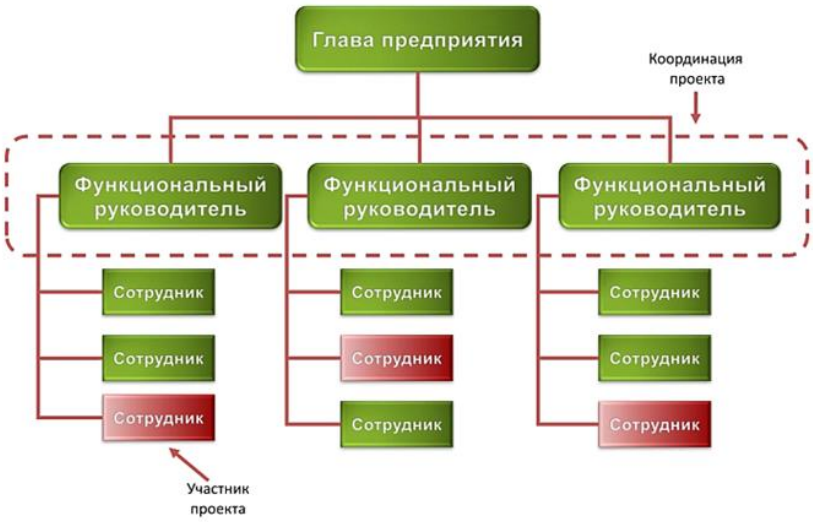

Рис.6-42. Функциональная структура компании [6-7].

Такая структура обеспечивает преемственность опыта разработчиков, рассчитана на стабильные отношения, однако не способствует коммуникациям в проекте и его управляемости. Альтернативой является **проектная структура,** в которой структурные подразделения создаются под проект (рис.6-43).

Рис.6-43. Проектная структура компании [6-7].

Она в большей степени соответствует требованиям отдельного проекта (коммуникации, управление), но не обеспечивает стабильности организации и преемственности опыта разработки.

Как у других. Большинство репертуарных театров имеют функциональную структуру: в оперном театре это - дирекция, отдел распространения, бутафорский и костюмерный

цеха, хор, оркестр, балетная труппа, оперная труппа. Проект (спектакль) не имеет собственной самостоятельной организационной структуры, все его финансовые и имиджевые показатели включаются в общий котел. Проектной формой театральной деятельности является *антреприза,* в которой актеры набираются под отдельный проект, имеющий самостоятельное финансирование и организационную структуру.

Матричная структура пытается в разных вариантах совместить интересы проекта и предприятия. Функциональная структура предприятия сохраняется, но степень независимости проекта может быть разной:

- *слабая матрица* - координатор проекта выполняет вспомогательные функции мониторинга, а управление проектом закреплено за функциональным руководителем;
- *сбалансированная матрица* - менеджер проекта функциональный руководитель имеют примерно равные полномочия - двойное подчинение;
- *сильная матрица* - при сохранении функциональной организационной структуры исполнители организованы в проектные команды, имеется отдельное подразделение менеджеров проекта.

Гибкие методологии требуют отдельных организационных решений. В п.5.2 уже рассматривалось *масштабирование Scrum* для крупных проектов в двух аспектах:

- масштабирование структуры исполнения проекта - объединение лидеров команд;
- масштабирование планирования проекта - объединение бэклогов и владельцов продуктов.

Для гибких технологии наиболее приемлемы:

- проектная структура организации;
- отдельная команда как самостоятельная хозяйственная единица;
- матричная структура организации - сильная матрица SCRUM of SCRUM;
- специфичные приемы планирования и самоорганизации при организации команды, метрики и планирования.

Фаза исследования. Скорость принятия решения

> *Скорость принятия решения - наибольшая скорость разбега многодвигательного самолёта, при которой в случае отказа двигателя возможно как безопасное прекращение, так и безопасное продолжение взлёта.* ***Энциклопедия «Авиация».***

В какой-то момент обсуждения проекта должно быть принято ключевое решение «Взлетаем». Естественно, что оно принимается руководством на основании вызывающих доверие данных. В унифицированном процессе для этих целей выделена первая фаза исследования, результатом которой является набор артефактов, на их основе принимается указанное решение (рис.6-44).

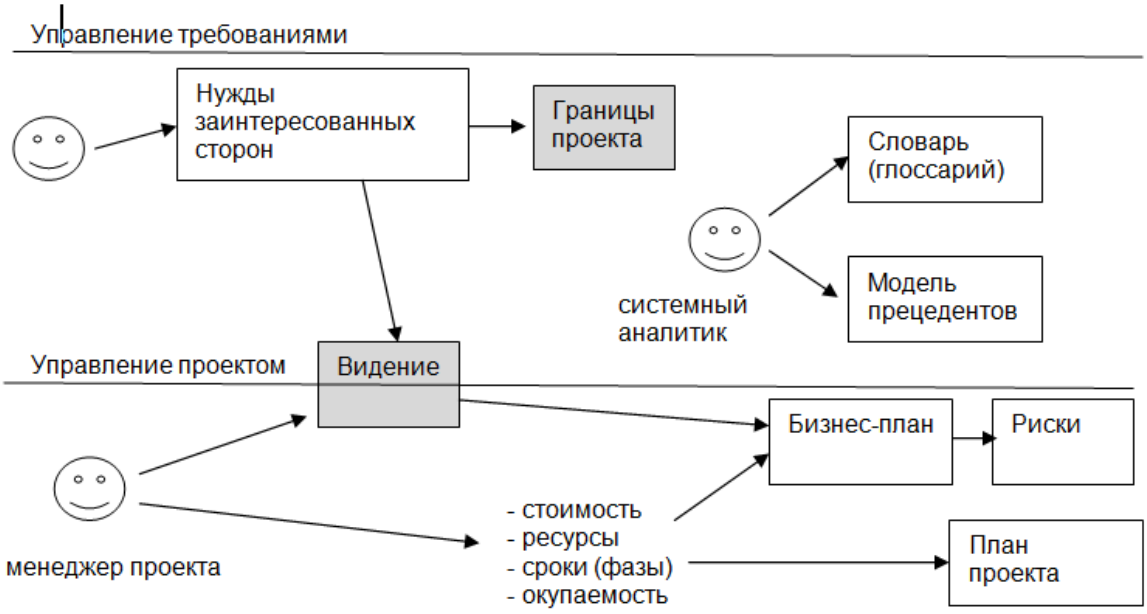

Рис. 6-44. Артефакты фазы исследования проекта

Главными технологическими процессами этой фазы являются *управление требованиями* и *управление проектом*. Выделим в структуре фазы исследования (см.5.1) компоненты, имеющие прямое или косвенное отношение к управлению проектом:

Цели фазы исследования:

- область применимости, граничные условия, видение проекта**;**
- предварительная оценка стоимости;
- предварительная оценка рисков;

Артефакты фазы исследования:

- документ видения;
- начальный бизнес-план;
- начальная оценка рисков;
- план проекта – фазы и итерации.

Деятельности фазы исследования:

- формулировка важнейших требований - **области действия проекта** и ограничении – **рамок проекта;**
- разработка **бизнес-плана** - оценка рисков, кадровое обеспечение, цена, график работ, рентабельность;

Риск – не благородное дело, а фактор разработки

Риск - неопределенное событие или условие, наступление которого отрицательно или положительно сказывается на целях проекта. Как правило, в случае возникновения негативного риска, почти всегда стоимость проекта увеличивается и происходит задержка в выполнении расписания проекта.

Приведенное определение ничего не говорит о сущности самих событий. Здесь сразу надо поставить жесткую границу. Если негативным событием нельзя управлять, то оно относится к *непрогнозируемым рискам,* единственной защитой от которых является создание *резерва* проекта или его *страхование.* Управляемые (прогнозируемые) риски также ограничиваются технологической и организационной сферами проекта. Таким

образом, рисками становятся обычные повседневные недостатки организации проекта, если с ними не мириться, а пытаться их контролировать. В доказательство перечислим перечни наиболее распространенных рисков от разных авторов:

Риски программного проекта по Боэму [6-19]:

- дефицит специалистов;
- нереалистичные сроки и бюджет;
- реализация несоответствующей функциональности;
- разработка неправильного пользовательского интерфейса;
- золотая сервировка, перфекционизм, ненужная оптимизация;
- непрекращающийся поток изменений;
- нехватка информации о внешних компонентах окружения системы;
- недостатки в работах, выполняемых внешними ресурсами;
- недостаточная производительность получаемой системы;
- разрыв в квалификации специалистов разных областей знаний.

Наиболее существенные риски по Демарко и Листеру [6-17]:

- изъяны календарного планирования;
- текучесть кадров;
- раздувание требований;
- нарушение спецификаций;
- низкая производительность.

Наиболее существенные риски по Архипенкову [6-7]:

- требования заказчика отсутствуют, подвержены частым изменениям;
- отсутствие необходимых ресурсов и опыта;
- отсутствие рабочего взаимодействия с заказчиком;
- неполнота планирования. забытые работы;
- ошибки в оценках трудоемкостей и сроков работ.

Обычно при первоначальной оценке объемов и сроков проекта обращают внимание на основную функциональность, в результате за бортом оказываются важный функционал и работы, связанные со вспомогательными технологическими процессами. Наиболее часто выпадают из первоначальной оценки:

- *функциональные требования*: программы установки, настройки, конфигурации, миграция данных, интерфейсы с внешними системами, справочная система и документация;
- *общесистемные требования*: производительность, надежность, открытость, масштабируемость, безопасность, кросплатформенность, эргономичность;
- *деятельности*: обучение, координация работ, уточнение требований, управление конфигурациями, управление версиями, автосборка, разработка автотестов, создание тестовых данных, обработка запросов на изменения;
- *накладные расходы*: сопровождение действующих систем, повышение квалификации, участие в подготовке технико-коммерческих предложений, участие в презентациях, административная работа, отпуска, праздники, больничные.

Резюмируя перечисленное многообразие, можно определить типичные места в процессе разработки, являющиеся источниками рисков:

- управление требованиями, функционал системы;
- планирование и оценка работ, оценка объемов и сроков;
- связь с внешними системами и информация о них;
- квалификация, опыт и взаимодействие исполнителей.

Для того, чтобы риском управлять, его необходимо определить *как артефакт жизненного цикла* со своим набором характеристик, например:

- причина или источник;
- симптомы риска;
- вероятность наступления риска;
- последствия риска, его *тяжесть* - влияние риска на возможность достижения целей проекта и на его основные параметры - стоимость, график, технические характеристики продукта.

Качественный анализ рисков

Поскольку большинство характеристик риска сложно определить количественно, остается экспертный, т.е. качественный подход по шкале *низкий-средний-высокий*. По этой шкале можно оценить две основных характеристики риска – **вероятность наступления** и **тяжесть последствий (воздействий)**. Обобщенный **показатель риска** можно определить как произведение *вероятность*воздействие***,** переведя качественные оценки в числовые значения. Тогда получим:

- исходные характеристики: 1 – низкий, 2-средний, 3 –высокий;
- показатель риска: 1-2 – низкий, 3-4 – средний 6-9 – высокий.

Управление рисками

Управление рисками является отдельной деятельностью в управлении программным проектом. Как и любая деятельность, она имеет свой план, в котором определяются:

- подходы (методология), инструменты и источники данных - метрика проекта;
- персоналии и ответственность;
- ресурсное обеспечение - учет в базовом плане по стоимости проекта;
- процесс - виды операций и их частота, привязанные к расписанию проекта;
- категории и классификация рисков, процедура идентификации.
- оценка вероятностей и степени влияния на параметры проекта - стоимость, сроки исполнения;
- источники данных о рисках: внутренние - статистика организации, экспертные оценки, внешние - отраслевая статистика, аналитика.

Результатом всего этого является определение варианта реагирования на каждый идентифицированный риск:

- **уклонение от риска** – изменение плана работ с целью исключения риска, например, приобретение стороннего ПО, привлечение специалиста;
- **передача риска** – передача ответственности за наступление риска на другую сторону, например, страхование ответственности или передача заказчику по условиям договора;
- **снижение рисков** – снижение вероятности наступления или возможных последствий риска, попытки снижения имеющейся неопределенности на более ранних стадиях проекта;

- принятие риска - пресловутое *авось*.

В целом стратегия проста: если риск не удается снизить административным путем, переложив ответственность, то необходимо попытаться уклониться от него. Если ее получается, как можно раньше снизить связанную с ним неопределенность, чтобы внести необходимые коррективы в оценку проекта.

Инструментом снижения рисков является прототипирование (рис.6-45), которое обеспечивает:

- снижение рисков по неопределенности требований путем использования функционального прототипа (макета);
- снижение архитектурных рисков и идентификация узких мест путем использования прототипа базовой архитектуры.

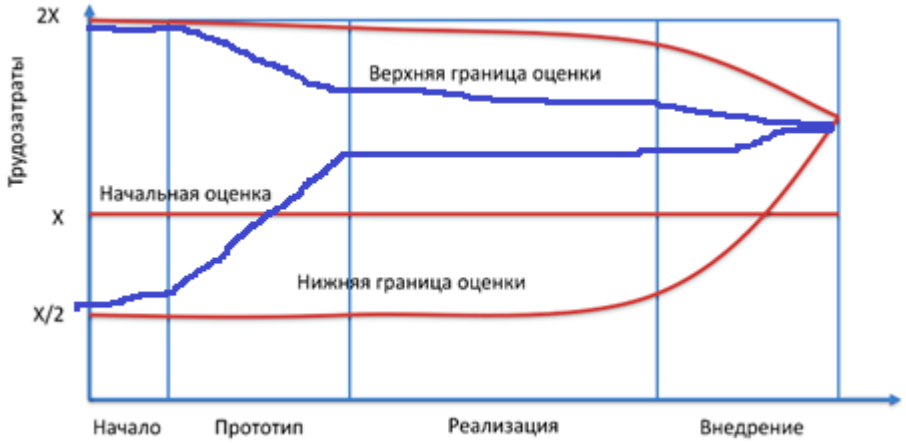

Рис.6-45. Влияние прототипирования на оценку затрат

Планирование проекта

Начальный этап планирования проекта включает в себя:

- уточнение содержания и состава работ. Результирующий артефакт - развернутое содержание в виде **иерархической структуры работ (ИСР) (Work Breakdown Structure, WBS)**;
- планирование организационной структуры, **формирование команды**;
- **базовое расписание проекта** – компоненты ИСР расписываются по срокам с учетом их взаимосвязей, занятости исполнителей и прочих внешних и внутренних факторов, например, сроков поставки оборудования.

Для внутреннего планирования проекта необходимо определить организационные структуры и регламенты для поддержки следующих видов деятельности:

- планирование управления содержанием – определение процедур работы с изменениями, вносимыми в проект (технологический процесс управления требованиями):
  - классификация объектов изменений: требования, архитектура, форматы и структуры данных, исходные коды, сценарии тестирования, документация;
  - определение источников запросов на изменение;
  - разработка порядка анализа, оценки и утверждения или отклонения изменения содержания;
  - определение порядка документирования изменений содержания;
- планирование управления конфигурациями:
  - хранение исходного кода;

- управление версиями;
- поддержание инфраструктуры ;
- администрирование БД;
- управление документами;

- планирование управления качеством:
  - принятие стандартов качества программного кода и документации - стилистические и технологические требования;
  - мониторинг качества: определение отклонений по качеству, причин и мер по их устранению, контроль исполнения;
  - аудит качества: независимая информация о несоответствиях, не устраняемых на уровне проекта.

Средством визуализации для базового расписания проекта является диаграмма Ганта (рис. 6-46) – графическое представление последовательности работ с учетом их причинно-следственной связи и распределением по исполнителям. Срок исполнения проекта по диаграмме Ганта ограничивает *критический путь* – самая длинная цепочка зависимых работ.

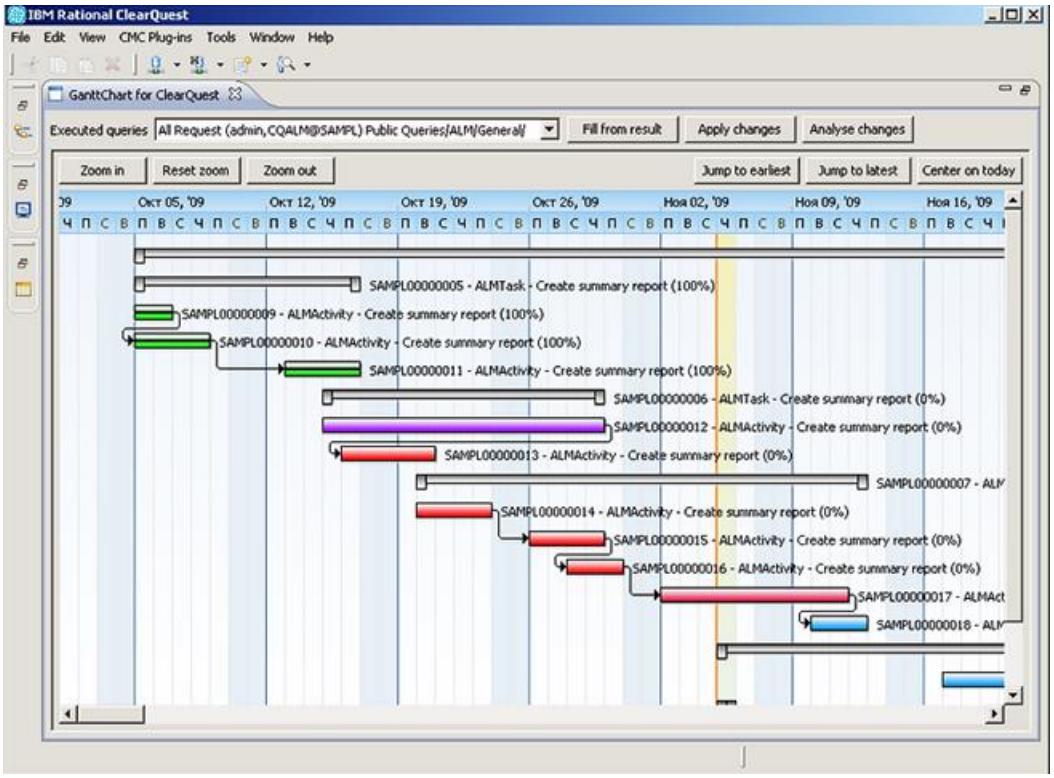

Рис.6-46. Диаграмма Ганта для взаимосвязанных работ.

При организации взаимосвязанных работ нужно учитывать вероятностный характер планирования [6-20] (рис.6-47). Если срок исполнения работы является случайной величиной, то срок исполнения проекта не будет равен простой сумме средних значений времени исполнения работ.

Рис. 6-47. Распределение сроков завершения взаимосвязанных работ

Обычно досрочное исполнение работы все равно может потребовать завершения синхронизированных с ним, а затягивание сроков автоматически сдвинет строки зависимых от него.

Стоимость и сроки

«Потом знакомые рекомендуют некоего Вавилыча, кристального старика. Кристальный старик приходит, с хватающей за душу медлительностью осматривает комнаты и берется сделать работу со своей олифой и кроном - все за двенадцать рублей. Тут же выясняется, что почтенный старец смертельно пьян и за свои слова отвечать не может. Его с трудом выводят.»

Ильф И., Петров Е. На купоросном фронте

Ключевой вопрос в момент принятия решения – сколько это будет стоить и в какие выльется сроки. В управлении программными проектами (УПП) эти риторические вопросы звучат так:

- трудоемкость проекта, на уровне проекта обычно измеряется человеко-месяцах (**N**);
- срок разработки в месяцах;
- количество исполнителей.

Результаты оценки являются определяющими для других деятельностей в УПП: планирование работ, подбор команды. Основным параметром для проекта является его трудоемкость, остальные параметры выводятся из него. Так, оптимальный **срок исполнения проекта** в месяцах при отсутствии других оснований рассчитывается из трудоемкости по формуле Боэма (см. методику COCOMO) как $\mathbf{T = 2{,}5\ N^{1/3}}$.

То, что для проекта определенного объема существует оптимальный срок его разработки, имеет под собой практические и житейские доводы:

- увеличение сроков приводит к увеличению трудоемкости: *работа занимает все отведенное для нее время* (законы Мэрфи);
- попытка форсирования сроков за счет увеличения числа исполнителей, наоборот, приводит к резкому увеличению трудоемкости за счет затрат на вхождение в проект, коммуникаций между участниками. Сроки при этом сильно не сокращаются, а, наоборот, могут вырасти.

Сметный подход – PERT

Смета – способ подсчета затрат как суммы произведений объемов отдельных работ, ресурсов и материалов на стоимость единицы. В своем натуральном виде он, естественно, не пригоден. Но если в качестве единиц работ использовать *количественные параметры* кода или функционала проекта, то такую «смету» можно использовать для оценки при определенных условиях:

- проект однороден, содержит значительное количество идентичных компонент – бизнес-объектов, форм, таблиц БД, событий, отчетов;
- выбранные параметры более-менее равномерно перекрывают программный код, т.е. затраты на реализацию одинаковых компонент приблизительно совпадают;
- проект идентичен по сложности и структуре с выполняемыми ранее командой проекта, имеется достоверная метрика по трудоемкости отдельных компонент.

Методика **PERT (Program / Project Evaluation and Review Technique)** учитывает вероятностный характер оценки на основе вероятностной оценки трудоемкости компонент каждого вида, но ничего не говорит о выборе этих компонент, оставляя это на совести оценщика. Сущность методики:

- определяется **n** видов функциональных параметров, имеющих количественную оценку;
- для каждого **i**-го типа вводится три вида трудоемкости единицы параметра: пессимистическая, средняя и оптимистическая - $\mathbf{P_i}$, $\mathbf{M_i}$, $\mathbf{O_i}$;
- вероятностная оценка (среднее и среднее квадратичное отклонение) по **i**-ой компоненте складывается из оптимистичной, средней и пессимистичной оценок как

  $$\mathbf{E_i = (P_i + 4Mi + O_i)/6}$$

  $$\mathbf{СКО_i = (P_i - O_i)/6}$$

- полная оценка получается как сумма по всему набору компонент как

  $$\mathbf{E = \Sigma E_i}$$

  $$\mathbf{СКО = (\Sigma СКО_i^{\,2})^{1/2}}$$

- оценка того, что трудоемкость с вероятностью 95% не превысит расчетную, определяется как $\mathbf{E_{95\%} = E + 2 * СКО}$

Замечание по теме. Обычно трудоемкость функциональных компонент оценивается по затратам на их кодирование, учитывая работу программиста, т.к. это легче всего сделать в метриках проекта. Тогда для полной оценки по всем технологическим процессам (дисциплинам) нужно учесть долю затрат на конструирование, которая составляет около 25%, т.е. окончательный результат умножается на 4.

Проект средней руки. Для однородного проекта, состоящего из набора приложений на основе клиент-серверной архитектуры с общим кодом для нижних уровней массовые компоненты реализации могут считаться типовыми при имеющейся метрике трудоемкости (рис.6-48).

Затраты чел/час	кол-во	P	M	O	Ei	СКОi	E	СКО
Приложений	5	30	20	10	20,0	3,33	100	17
Экранов (форм)	21	10	6	4	6,3	1,00	133	21
Элем. Управления	190	5	3	1	3,0	0,67	570	127
Отчетов	12	20	10	4	10,7	2,67	128	32
Событий	130	5	3	1	3,0	0,67	390	87
Таблиц БД	17	12	5	3	5,8	1,50	99	26
Полей БД	90	3	2	1	2,0	0,33	180	30
Бизнес-объектов	15	15	5	3	6,3	2,00	95	30
Форматов файлов	7	10	8	5	7,8	0,83	55	6
						Итого	**1750**	**167**
						E 95%	**2083**	
						E ч/мес	**51**	
		=H26/165/0,25				**Срок**	**9,2**	
						Исп	**5,5**	

Рис. 6-48. Оценка по методике PERT(чел*час)

Основным недостатком сметного подхода является то, что он ориентирован на метрики конкретного технологического процесса в команде разработчиков. Метод функциональных точек [6-7] пытается охарактеризовать продукт с более общих позиций. Функциональная точка (FP) – обобщенная единица функциональности продукта. Различные компоненты функциональности нормируются, учитываются различные факторы, влияющие на проект аналогично методике COCOMO. Трудоемкость в выровненных функциональных точках перечитывается в строки кода (объем проекта) в соответствии с языком программирования.

Отраслевая статистика - COCOMO

Другой способ определения трудоемкости основан на анализе отраслевой статистики. Имея статистику основных параметров проектов - объемные показатели, трудоемкость, сроки, а также дополнительных факторов, оказывавших влияние на проекты, можно построить статистические зависимости трудоемкости и сроков от объемных показателей проекта. Наиболее известна методика **COCOMO – COnstructive COst MOdel** [6-21] создана на основе анализа данных 161 проекта в области ИТ в 1997 г.. Несмотря на свой солидный математический базис, она также технологически непродуктивна, т.к. опирается на единственный входной объемный показатель **SLOC** – количество строк кода в тысячах (*кило-строк*). Это значение легко получить для готового проекта задним числом, а на этапе исследования придется опять-таки использовать оценку объема проекта в SLOC разными способами:

- на основе общих оценок *масштаба* проекта;
- используя тот же *сметный подход*, только вместо трудоемкости оценивать объемные показатели кода SLOC в среднем на единицу компоненты каждого вида.

Итак, методика COCOMO использует модель, полученную на основе анализа статистики трудоемкости проектов методом *регрессионного анализа*. *Регрессия* – это интерполяция статистических данных линейной или экспоненциальной зависимостью. В данном случае используется экспоненциальная регрессия вида $\mathbf{Y=a\ X^b}$. Аргумент функции регрессии – **SLOC** – объем проекта в *кило-строках* кода.

Имеется три варианта моделей и три уровня оценки - **базовый, средний и детальный**. Базовый уровень выглядит так:

- трудоемкость $\mathbf{V = a \cdot SLOC^{\,b}}$ [человеко-месяцев];

- срок разработки или длительность $\mathbf{T = c \cdot V^{d}}$ [месяцев];
- число разработчиков $\mathbf{N = V / T}$ [человек];

Коэффициенты a,b,c,d определены для разных классов проектов**:** *органический, полуразделенный, встроенный* – в зависимости от уровня и жесткости требований (рис.6-49).

Труд	Срок	Исп		a	b	c	d
43	10	4	однородный	2,4	1,05	2,5	0,38
64	11	6	разнородный	3	1,12	2,5	0,35
96	11	9	встроенный	3,6	1,2	2,5	0,32
ч/мес	мес	чел					

Рис.6-49. Базовый уровень оценки COCOMO II

Отсюда следует знаменитая формула Боэма для расчета сроков проекта - $\mathbf{T = 2.5\ V^{1/3}}$.

Модель *среднего уровня* учитывает поправки, вносимые различными факторами, одни из которых являются линейными множителями - *множители трудоемкости*, а другие добавляются к показателю степени - *факторы масштаба*:

$$\mathbf{V = a \cdot KLOC^{b} \prod F_i}$$

$$\mathbf{a = 2.94\ \ b = 0.91 + 0.01 \sum M_j}$$

где, $\mathbf{F_i}$ – множители трудоемкости, $\mathbf{M_j}$ – факторы масштаба. Значения коэффициентов заданы в таблицах в зависимости от *качественного состояния фактора*.

Множители трудоемкости $\mathbf{F_i}$:

1. PERS – квалификация персонала/ Качественное состояние фактора: аналитики и программисты имеют низшую квалификацию, текучесть больше 45% - аналитики и программисты имеют высшую квалификацию, текучесть меньше 4%;
2. RCPX – сложность и надежность продукта;
3. RUSE – разработка для повторного использования;
4. PDIF – сложность платформы разработки;
5. PREX – опыт персонала;
6. FCIL – оборудование;
7. SCED – сжатие расписания. Диапазон изменения фактора от 75% до 160% от номинальной длительности.

Факторы масштаба $\mathbf{M_j}$ **:**

1. PREC – прецедентность, наличие опыт аналогичных разработок;
2. FLEX – гибкость процесса разработки;
3. RESL – архитектура и разрешение рисков;
4. TEAM – сработанность команды;
5. PMAT – зрелость процессов.

Модель *детального уровня*, выполненная в виде приложения – калькулятора [6-21] использует расчетные формулы для многокомпонентного (многомодульного) проекта. Она подсчитывает вероятностные оценки трудоемкости - оптимистическую, пессимистическую, наиболее вероятную (рис.6-50), а также трудоемкость по модулям, фазам и технологическим процессам (рис.6-51).

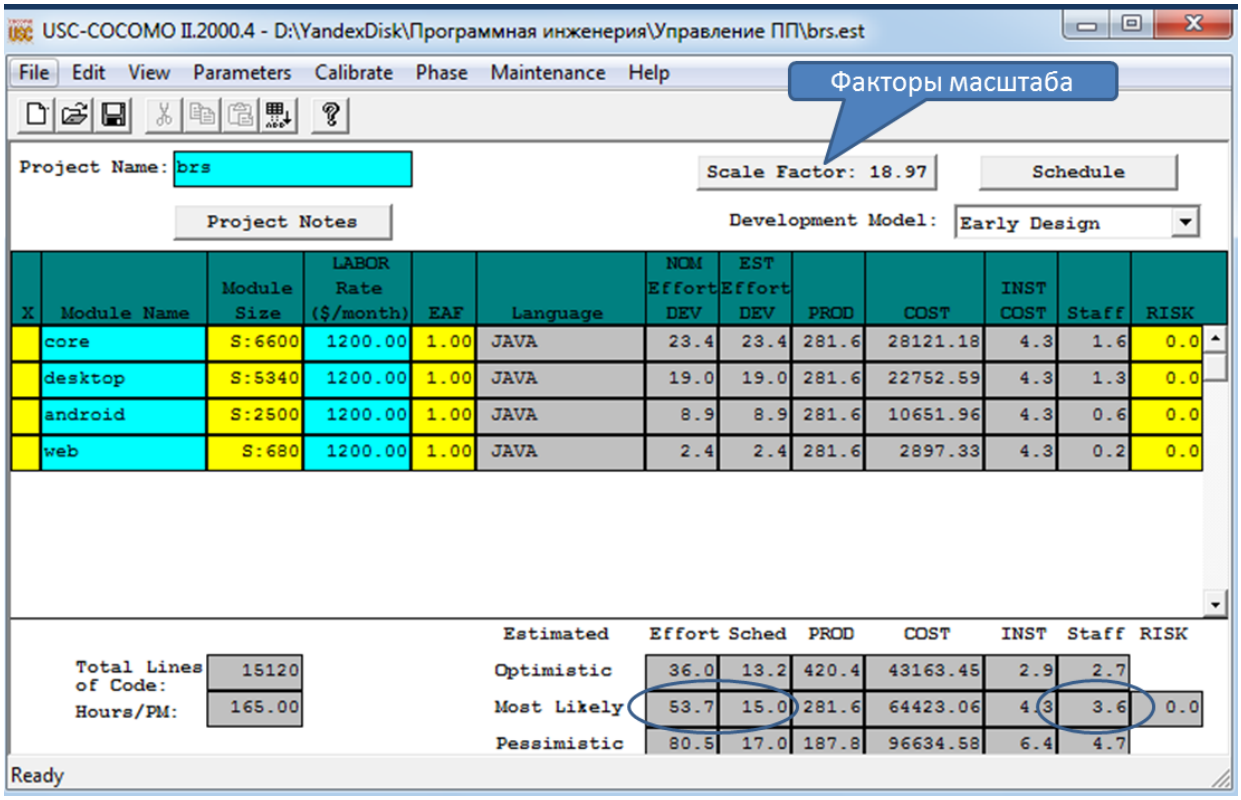

Рис.6-50. Калькулятор трудоемкости COCOMO II для многокомпонентного проекта

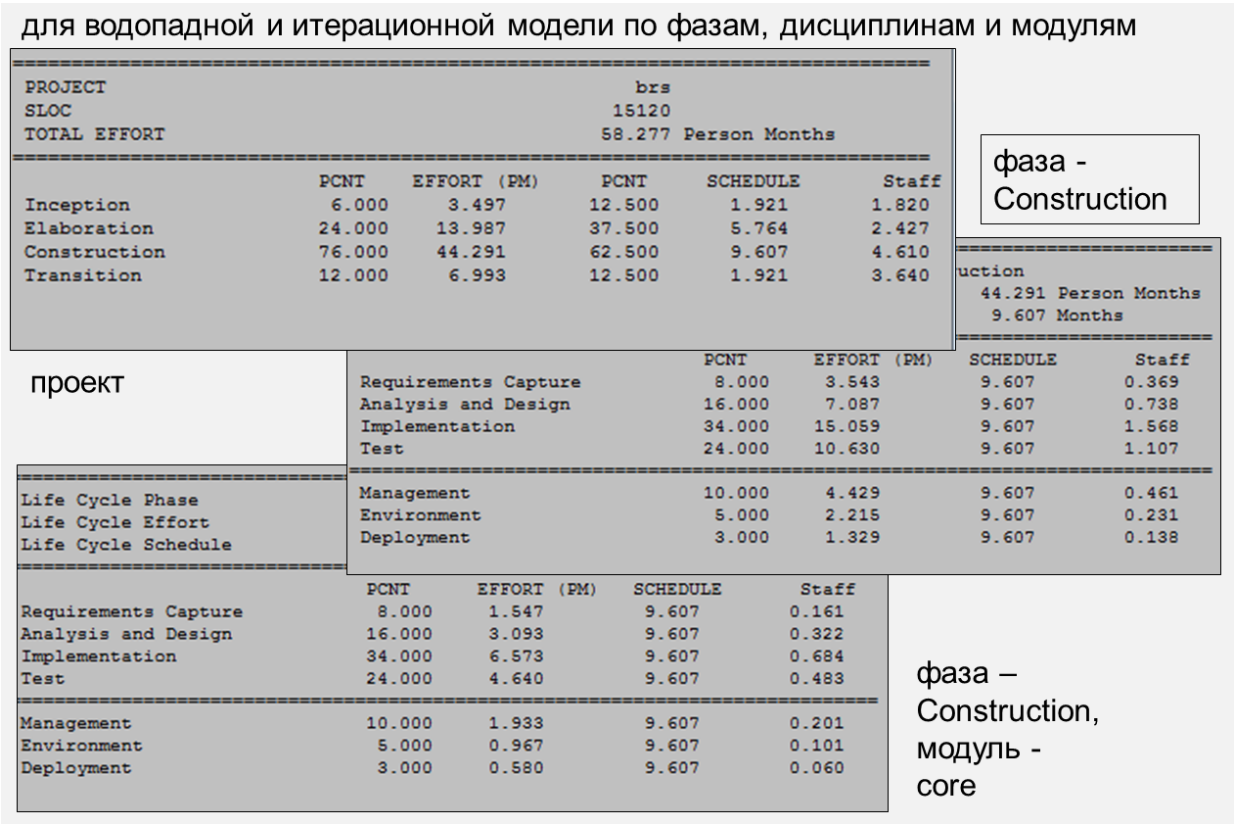

Рис.6-51. Расчет детального уровня COCOMO II по фазам, дисциплинам и модулям

Как у других. Сметный подход широко используется в *градостроительстве*, где он регламентируется **СНИП**ами (**С**троительные **Н**ормы **И** **П**равила). На их основании производятся расчеты материальных затрат на строительство, исходя из основных параметров зданий, сооружений, автодорог и пр.. При этом используются поправочные коэффициенты для различных внешних факторов и условий эксплуатации. Отраслевая статистика не может быть основанием для расчета, т.к. при соблюдении СНИПов она является их следствием.

Формирование команды

Структура команды исполнителей проекта определяется в соответствии с деятельностями и дисциплинами унифицированного процесса. В самом общем случае она состоит из нескольких групп.

1. Анализ. Исполнители фазы исследования и дисциплины управления требованиями:
   - **бизнес-аналитик.** Построение модели предметной области;
   - **бизнес-архитектор**. Разработка бизнес-концепции (видения) системы;
   - **системный аналитик**. Перевод требований к продукту в функциональные требования к ПО;
   - **специалист по требованиям**. Документирование и сопровождение требований к продукту;
   - **менеджер продукта** (функциональный заказчик). Представление в проекте интересы пользователей продукта.
2. Управление проектом и функциональная административная верхушка:
   - руководитель проекта;
   - куратор проекта. Оценка планов и исполнения проекта, выделение ресурсов;
   - системный архитектор, архитектор ПО. Разработка технической концепции системы, принятие ключевых проектных решений;
   - руководитель группы тестирования;
   - ответственный за управление изменениями, конфигурациями, за сборку и поставку программного продукта.
3. Производство. Проектирование, конструирование, сопровождение ПО:
   - **проектировщик.** Проектирование компонентов и подсистем в соответствие с общей архитектурой, разработка архитектурно значимых модулей;
   - проектировщик БД;
   - проектировщик графического интерфейса;
   - разработчик.
4. Тестирование:
   - **проектировщик тестов**. Разработка тестовых сценариев;
   - **разработчик автоматизированных тестов**;
   - **тестировщик**. Тестирование продукта. Анализ и документирование результатов.
5. Обеспечение. Инструментальная поддержка. Вспомогательные процессы:
   - технический писатель;
   - переводчик;
   - дизайнер графического интерфейса;
   - разработчик учебных курсов, тренер;
   - участник рецензирования;

- специалист по продажам, маркетолог;
- системный администратор, технолог;
- специалист по инструментальным средствам.

Приведенный список содержит перечень исполнителей по максимуму. В зависимости от размера проекта и его содержания некоторые деятельности могут быть исключены или совмещаться одним исполнителем. Основания для совмещения могут быть разными. Например:

- системный архитектор, архитектор ПО с разработчиком – удачное совмещение, поскольку архитектор ПО по сути вырастает из разработчика;
- разработчик с дизайнером графического интерфейса – неудачное совмещение, от дизайнера требуются специфические знания художественного дизайна, а разработчик будет превносить в интерфейс свои внутренние представления о реализации.

Структура и команды исполнителей и график их включения в проект (рис.6-52) разрабатываются на основе следующих данных:

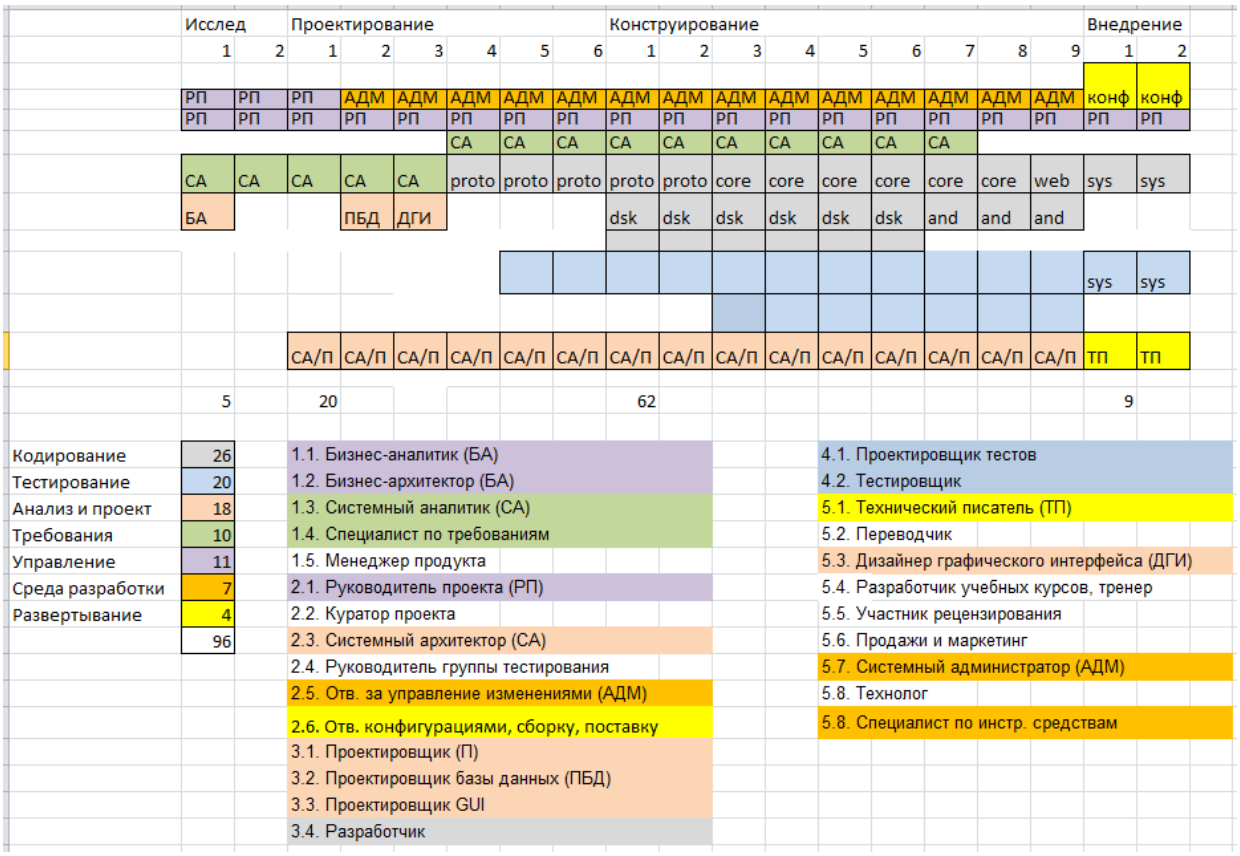

Рис. 6-52. Структура команды проекта

- общей трудоемкости проекта и сроков разработки;
- базового расписания проекта;
- распределения трудоемкости и сроков исполнения проекта по этапам (фазам) и технологическим дисциплинам, например, с использованием калькулятора трудоемкости из методики COCOMO-2 [6-21].