

Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана

baryshnikovam@mail.ru

Лекция 1

Основные понятия и особенности
промышленного рынка программных
продуктов. Классификация программных
продуктов. Экономика создания сложных
программных продуктов



Экономика программной инженерии

Лектор – Барышникова Марина Юрьевна

Преподаватели, ведущие лабораторные работы – Барышникова Марина Юрьевна, Силантьева Александра Васильевна (ауд. 513-л)

Структура курса:

Продолжительность курса – 10 недель

Лекции – вторник, 15-40, ауд. 224-л

Практические занятия: вторник

Контрольные мероприятия:

5 нед. – рубежный контроль – 35 б.

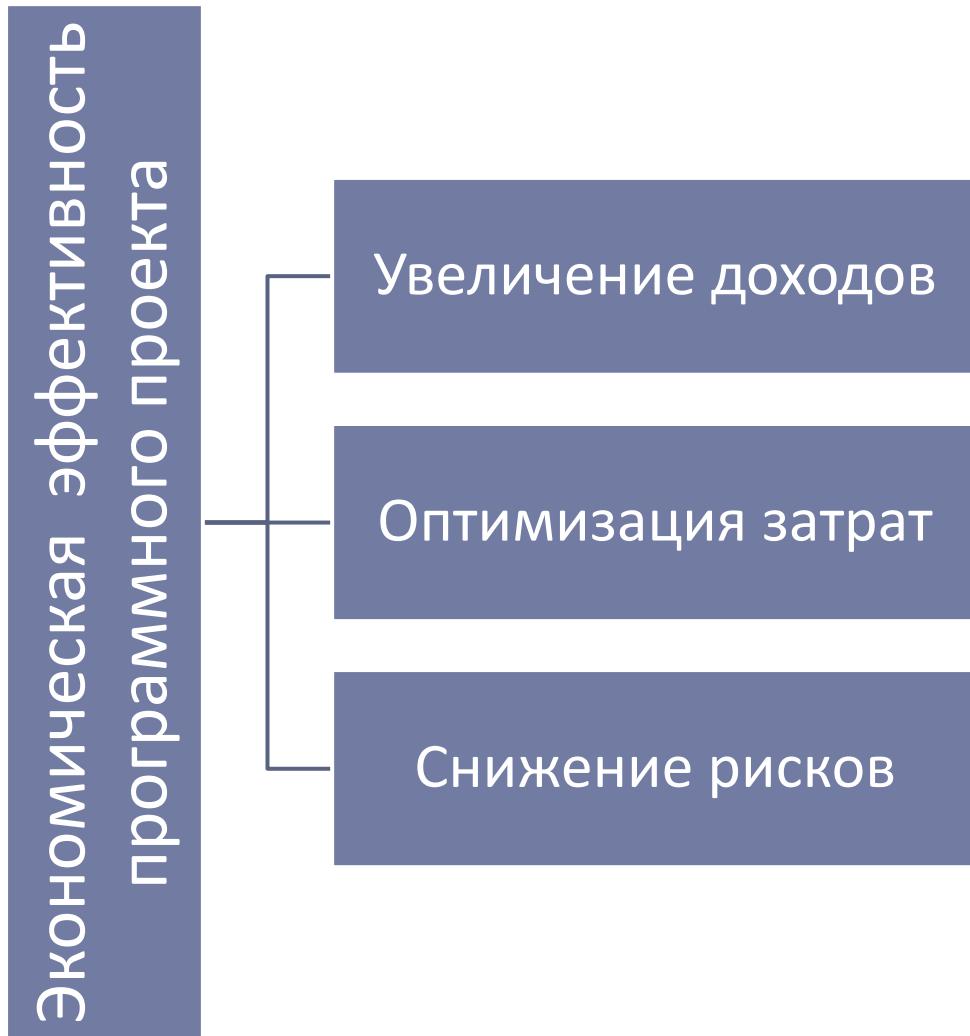
10 нед. – рубежный контроль – 35 б.

12-13 нед. – экзамен по курсу (максимум – 30 баллов)

**Информационный ресурс для поддержки курса:
[http://e-learning.bmstu.ru/portal_iu7/course/
view.php?id=9](http://e-learning.bmstu.ru/portal_iu7/course/view.php?id=9)**



Экономика программной инженерии: зачем и о чём?



В ходе реализации своей профессиональной деятельности разработчики информационных технологий (к числу которых относится и программное обеспечение) сталкиваются с рядом проблем:

- ▶ как грамотно, с учетом зарубежных и отечественных стандартов, организовать процесс разработки;
- ▶ как определить и обосновать трудозатраты на создание программной системы (ПС);
- ▶ как аргументировать для заказчика стоимость разработки;
- ▶ как, учитывая условия сложившегося рынка программных продуктов (ПП), обеспечить требуемый уровень рентабельности своего проекта

Экономика программной инженерии: зачем и о чём?

Рыночные
отношения,
возникающие в
ходе производства
и реализации ПО
как товара

- Производство ПО – динамично развивающаяся отрасль. Несмотря на то, что доля ИТ-отрасли в российском ВВП составляет примерно **2-3%**, это довольно перспективный сектор рынка, а также источник высокооплачиваемых рабочих мест
- Несмотря на пандемию, в 2020 году российский ИТ-рынок вырос на **14%** до уровня **1,83 трлн. руб.**, из которых **11%** составляет разработка ПО
- Самыми популярными направлениями в 2020 г. стали искусственный интеллект, облачные сервисы, приложения для удаленной учебы, работы и развлечений, решения для обеспечения информационной безопасности в распределенных корпоративных сетях

Экономика
создания ПО
промышленными
методами

- Проекты по созданию ПО промышленными методами должны начинаться с прогнозирования, анализа и технико-экономического обоснования (ТЭО)
- Экономическое прогнозирование проектов в программной инженерии должно опираться на использование достаточно точных методов для оценивания экономических характеристик производства сложных программных продуктов

РЫНОК – ЭТО...

Рынок – это система экономических отношений, складывающихся в процессе производства, обращения и распределения товаров. Это определенный способ согласования деятельности участников общественного производства; механизм, соединяющий производителя и потребителя на основе спроса и предложения; это саморегулирующаяся и самонастраивающаяся на спрос система

Субъектами рынка являются продавцы и покупатели. Субъекты взаимодействуют на рынке, образуя взаимосвязанный «поток» купли-продажи



Объектами рынка являются товары (услуги) и деньги

Товар – любой продукт производственно-экономической деятельности в материально-вещественной форме

Услуги – итог непосредственного взаимодействия поставщика и потребителя и внутренней деятельности поставщика по удовлетворению потребности потребителя



Российский ИТ-рынок: некоторые цифры

Крупнейшие ИТ-компании России 2021

№ 2020	Название компании	Совокупная выручка компании в 2020 г., с НДС, ₽тыс.	Рост выручки 2020/2019, в %	Штатная численность сотрудников в компании на 31.12.2020
1	Ланит	216 810 014	24,8%	14 100
2	OCS Distribution	214 991 706	н/д	2 300
3	EPAM Systems	191 322 847	28,9%	36 737
4	Марвел-Дистрибуция	156 139 390	60,1%	1 232
5	Softline	131 953 000	21,2%	н/д

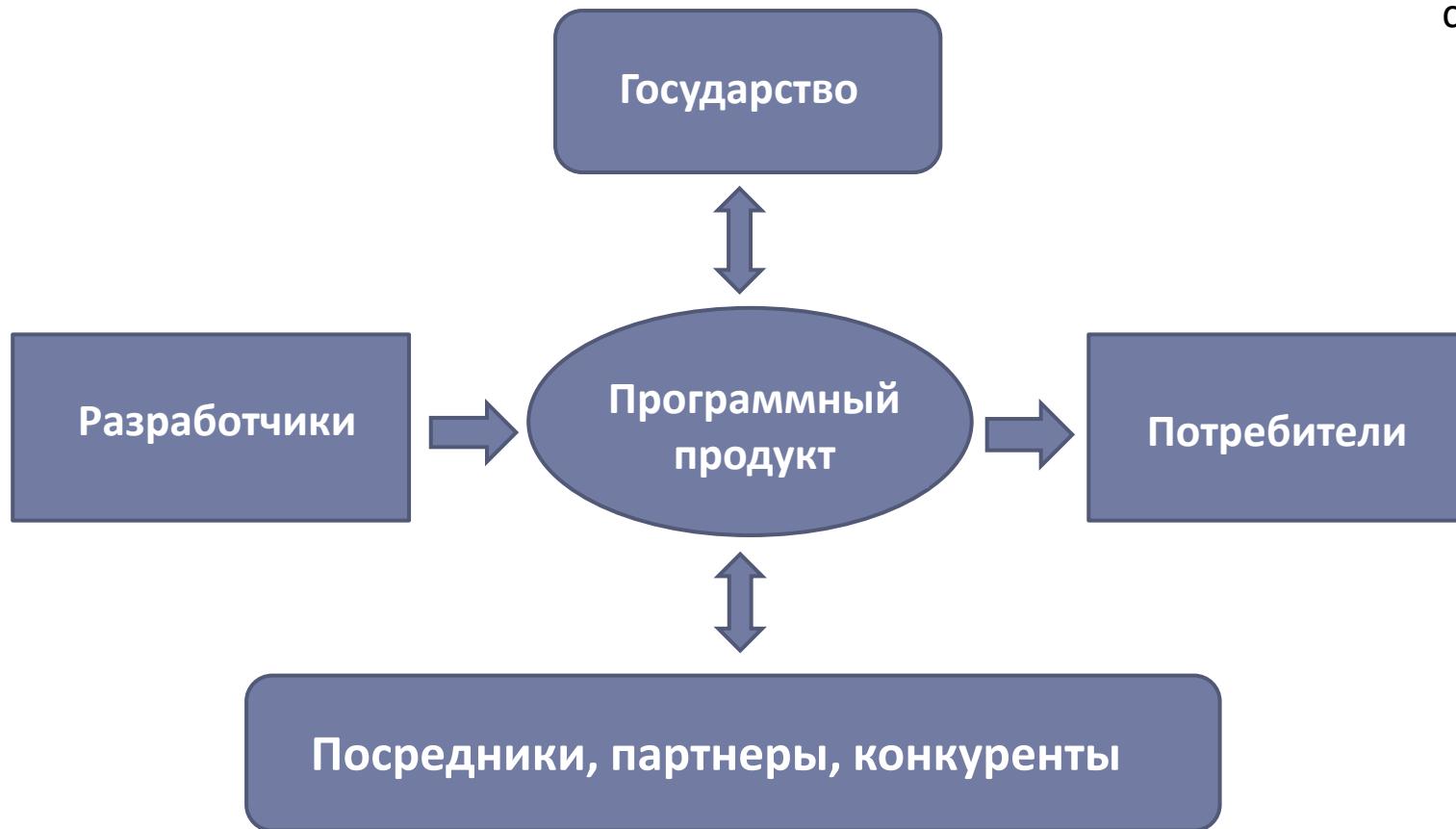
Источник: CNews Analytics, 2021

Российская ассоциация электронных коммуникаций (РАЭК) подсчитала, что в 2020 г. выручка российских интернет-компаний увеличилась на **22%** по сравнению с 2019 г. и достигла **₽6,7 трлн**. Из них **₽349,8 млрд** приходится на маркетинг и рекламу, **₽6,07 трлн** — на электронную коммерцию, **₽152,3 млрд** — на инфраструктуру, а **₽123,4 млрд** — на цифровой контент

По данным РАЭК в 2020 году кадровый дефицит в ИТ-секторе составил **150 тыс.** чел., к 2024 г. эта цифра должна увеличиться до **300 тыс.** чел.

Всего в апреле 2021 г. на портале hh.ru было представлено более **105 тыс.** вакансий в ИТ-сфере (в Москве – **35,8 тыс.**), из них более **41 тыс.** вакансий предлагалось для программистов и разработчиков ПО. На апрель 2021 г. средняя заработка разработчика ПО в Москве составляла **165 449 руб.**

Рынок программных продуктов и его участники



Рынок ПО существует при соблюдении следующих условий:

- наличие реальной потребности у конкретных потребителей (спрос)
- наличие конечных продуктов у производителей, ориентированных на удовлетворение потребностей в ПО (предложение)
- развитая сеть посредников между производителями и потребителями
- наличие экономических и организационно-правовых механизмов, регламентирующих цивилизованное взаимодействие участников

Разработчики (правообладатели)

Факторы, определяющие конкурентоспособность разработчиков и их положение на рынке:

- ▶ отличительные особенности продукции, побуждающие пользователя приобретать именно данный программный продукт
- ▶ цена на программную продукцию или услугу
- ▶ качество продукции с точки зрения удовлетворения существующих потребностей пользователей
- ▶ гибкость производителя, связанная со способностью реагировать на просьбы покупателя по адаптации либо доработке программного продукта
- ▶ время (сроки) реагирования производителя на потребности покупателя (время адаптации и внедрения ПП, продолжительность обучения пользователей, период гарантийного сопровождения, временные условия по модернизации и поставке новых версий и пр.)



Причины, препятствующие развитию рынка со стороны разработчиков

- ▶ Отсутствие начального капитала на развитие фирмы, наработку требуемых заделов, приобретение лицензионного ПО
- ▶ Слабое представление о существующем рынке конкурирующих программных продуктов
- ▶ Ориентация производителей на мелкосерийное производство программных продуктов, разрабатываемых обычно под конкретный заказ
- ▶ Высокая доля фиксированных затрат в структуре издержек и, как следствие, высокие цены на создаваемые программные продукты
- ▶ Использование при разработке пиратских инструментальных программных средств, не позволяющее производителю открыто рекламировать свои продукты, участвовать в выставках и пр.
- ▶ Отсутствие эффективных программных средств защиты от копирования, а также экономических и юридических механизмов, препятствующих этим процессам
- ▶ Отсутствие опыта по представлению ПП в виде законченного продукта и организации маркетинга по его распространению
- ▶ Отсутствие профессиональных менеджеров по продвижению ПП на рынок
- ▶ Незнание или несоблюдение отечественных и международных стандартов по управлению жизненным циклом, качеством и документированием ПП



Потребители

Экономические интересы потребителей, в качестве которых могут выступать государственные (муниципальные) структуры, а также юридические и физические лица, заключаются в приобретении рыночных преимуществ и доходов от использования программных продуктов, либо в удовлетворении в той или иной мере личных потребностей

Проблемы, с которыми сталкиваются потребители при приобретении ПП:

- ▶ отсутствие сформированного спроса на программное обеспечение и четкого представления о технологии использования программных продуктов в практической деятельности
- ▶ низкая информированность потребителей о рынке предлагаемого ПО
- ▶ неспособность четко сформулировать требования к приобретаемым программным продуктам, превышение значимости ценового критерия по сравнению с критерием качества при выборе приобретаемых ПП
- ▶ несоответствие между высокими ценами на программное обеспечение и сиюминутными «выгодами» от его использования
- ▶ незнание, а чаще игнорирование экономических и нормативно-правовых механизмов цивилизованной работы на рынке (ментальность отечественного потребителя не расценивает факт использования нелегальных копий как хищение собственности производителя)



Государство

- ▶ Осуществляет регулирование отношений, возникающих по поводу использования программного обеспечения, посредством создания экономических, организационных и нормативно-правовых механизмов, обеспечивающих цивилизованное взаимодействие участников рынка ПП
- ▶ Интересы государства заключаются, в первую очередь, в получении выгод от надлежащей охраны прав интеллектуальной собственности и повышении эффективности ее использования в интересах развития отраслей экономики

Недостатки государственной политики в сфере регулирования рынка ПП:

- ▶ имеющиеся законы об охране авторских прав, защите интеллектуальной собственности, информации, информатизации и защите информации практически не работают, так как нет эффективных механизмов их конкретного применения
- ▶ существующая система нормативных документов (ГОСТов), регламентирующих жизненный цикл проектирования и документирования программных средств морально устарела и носит рекомендательный характер
- ▶ сертификация как институт, обязывающий создавать программные продукты с определенными параметрами качества, существует преимущественно в добровольной форме и не носит масштабного характера



Посредники

В качестве посредников выступают фирмы, берущие на себя функции маркетинга и распространения программного продукта

Их роль заключается в принятии продукта от разработчика, оценке его готовности к выводу на рынок, осуществлении мероприятий по продвижению ПП и доведению его до конечного пользователя

В рамках этой деятельности разделение функций между разработчиками и посредниками является наиболее эффективным механизмом оптимизации усилий по доставке ПП пользователю, так как в этом случае с разработчика снимается нагрузка по исследованию рынка, рекламе, доставке и пр.

Партнеры

В качестве партнеров могут выступать фирмы, производящие аналогичную продукцию и ориентированные на один и тот же сегмент рынка

Направления сотрудничества:

- ▶ интеграция в сфере приобретения и совместного использования средств производства программного продукта
- ▶ освоение каналов распространения и активизация маркетинговой деятельности



Перспективные направления развития рынка ПО

- ▶ усиление роли государства в части нормативно-правового регулирования рынка, финансовой поддержки небольших коллективов в виде получения грантов, льготных кредитов, налоговых льгот на начальных этапах развития фирм
- ▶ развитие системы сертификации программных систем на соответствие отечественным и международным стандартам, в том числе, введение сертификации ПС в качестве обязательного условия при финансировании работ из бюджетов всех уровней
- ▶ повышение качества и эффективности рекламной деятельности, публикация материалов в популярных компьютерных журналах, участие в специализированных выставках-ярмарках, создание электронных каталогов и их рекламу в Internet
- ▶ развитие сети оптовой торговли программным обеспечением через сеть филиалов, работающих с фирмой-производителем на контрактной основе
- ▶ формирование потребности у пользователей в приобретении качественной, сертифицированной продукции



Программный продукт как товар



Программный продукт вступает в хозяйственный оборот как товар только в случае фиксации его на материальном носителе, однако обладание материальным носителем информации не делает его приобретателя уникальным собственником информации

Программный продукт - это самостоятельное отчуждаемое произведение, представляющее собой публикацию текста программы на языке программирования или в виде исполняемого кода

- ▶ является предметом интеллектуального труда
- ▶ охраняется авторским правом
- ▶ вовлекается в хозяйственный оборот либо посредством коммерциализации (купли-продажи, переуступки прав собственности), либо посредством капитализации (постановки на баланс, инвестирования в основной капитал)



Компьютерные программы как товар

- ▶ Программный продукт – совокупность записанных на носителях данных программных компонентов, являющихся результатом промышленного производства, предназначенных для поставки, передачи или продажи пользователю, снабженных технической документацией, инструкциями по обучению пользователей, а также гарантийными обязательствами по сопровождению и обслуживанию
- ▶ Программный модуль – отдельно компилируемая часть программного кода
- ▶ Программный компонент – это автономный элемент программного обеспечения, предназначенный для многократного использования, который может распространяться для использования в других программах в виде скомпилированного кода
- ▶ Программный комплекс (программная система) – набор взаимодействующих программ, согласованных по функциям и форматам, имеющих единообразные, точно определенные интерфейсы и составляющих полное средство для решения больших задач
- ▶ Коробочный программный продукт – программное обеспечение, предназначенное для неопределенного круга покупателей и поставляемое на условиях «как есть» со стандартными для всех покупателей функциями

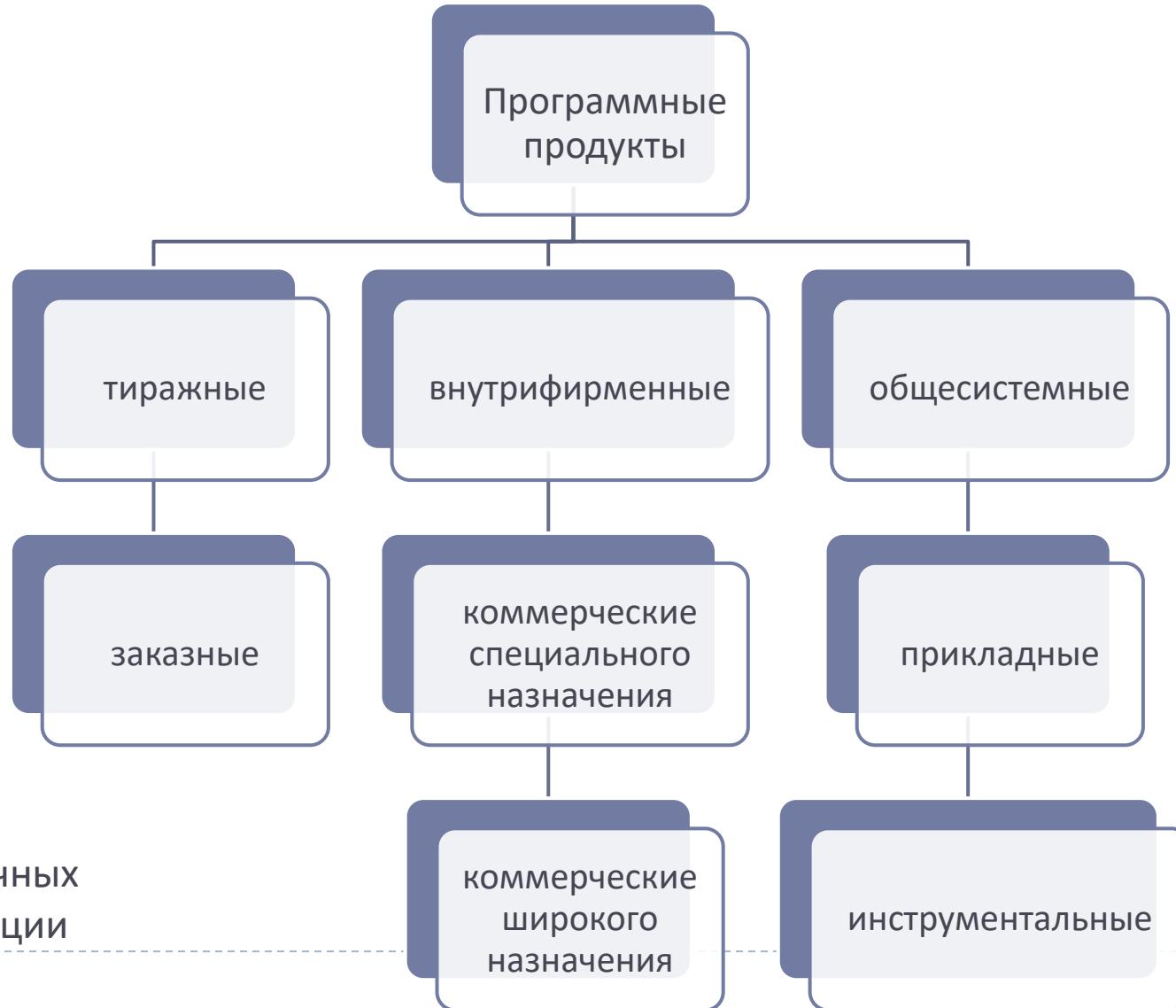


Особенности программного продукта как товара

- ▶ Нематериальная природа существования (ПП нельзя увидеть в процессе конструирования и, следовательно, оперативно повлиять на его реализацию)
- ▶ Возможность неоднократной продажи и участия одновременно в нескольких сделках
- ▶ Сохранение свойств продукта в процессе использования (не исчезает и не изнашивается)
- ▶ Создание продукта в условиях повышенного риска, невозможность точного оценивания временных и финансовых параметров разработки, обусловленная творческим характером труда в процессе интеллектуальной деятельности
- ▶ Относительно низкие затраты на тиражирование по сравнению с высокими затратами на разработку, обусловленные ничтожно малой стоимостью производственных операций на создание копий ПП

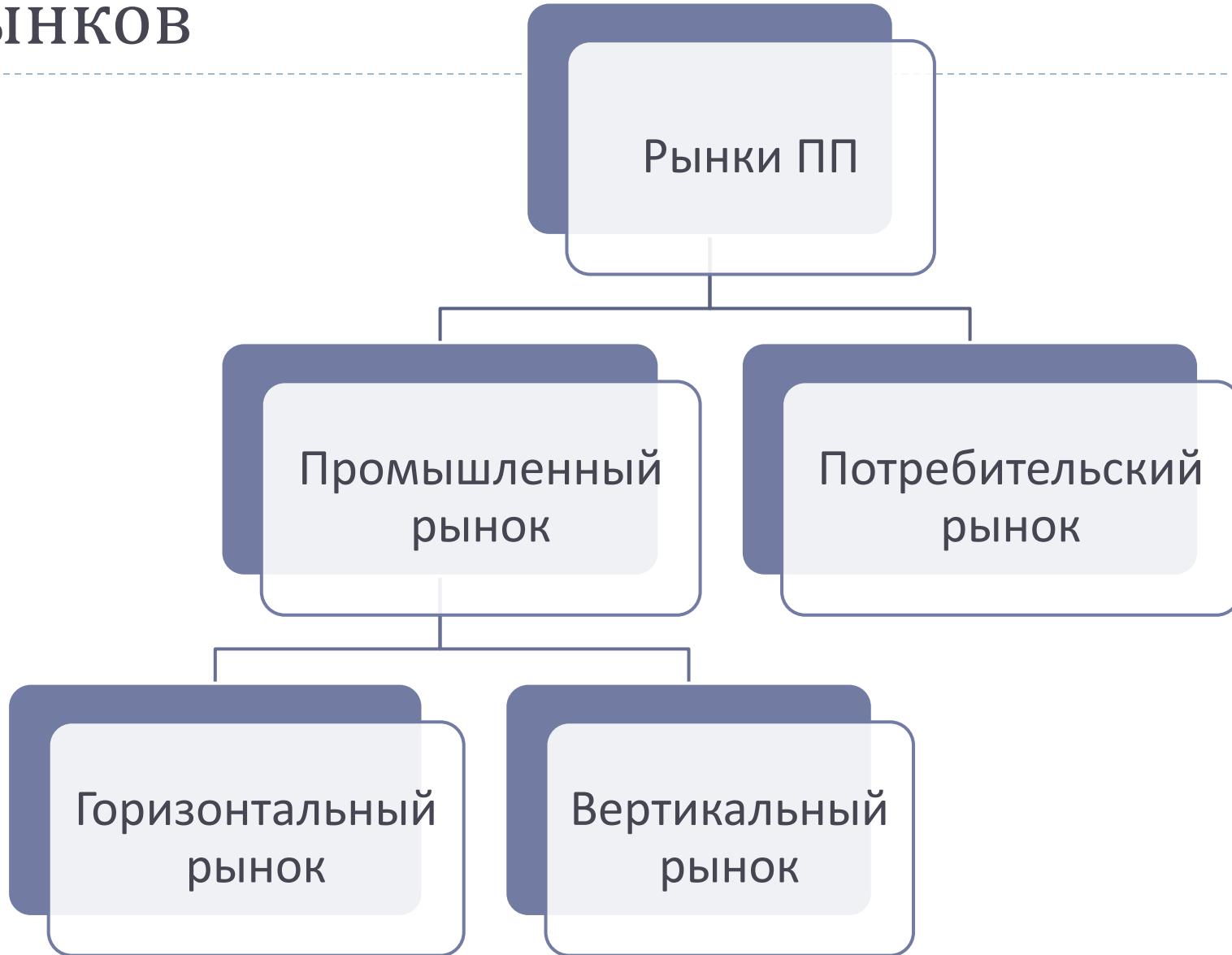


Классификация программных продуктов



Примечание: схема
объединяет три различных
основания классификации

Виды рынков



Потребительский рынок

Характеризуется наличием товаров и услуг для личного потребления, покупаемых или приобретаемых другим способом отдельными физическими лицами

Количество покупателей в данном сегменте рынка достаточно велико и все они рассматриваются как единая генеральная совокупность

Чаще всего покупатель не осведомлен об истинных характеристиках товара, а больше доверяет рекламе или продавцу-консультанту

Анализ поведения пользователей проводится на представительной выборке. По результатам анализа выявляется среднестатистический потребитель



Промышленный рынок (рынок корпоративных продаж)

Характеризуется наличием множества товаров промышленного назначения, которые могут использоваться как самостоятельно, так и в составе других товаров или услуг, продаваемых, сдаваемых в аренду или поставляемых другим потребителям

Количество участников рынка невелико. Производители и покупатели обладают высокими профессиональными навыками в области ПП, представленных на рынке, а также навыками коммерческой работы

Объектами переговоров и подписания контрактов с каждым конкретным заказчиком являются такие условия сделки как цена, качество, условия поставки и оплаты



Сегменты рынка корпоративных продаж

Горизонтальный рынок представляет собой совокупность различных изделий и/или услуг общего назначения и состоит из широкого спектра отраслей. Субъекты (пользователи) горизонтального рынка имеют потребности в решении проблем общего характера. ПП для горизонтального рынка создаются с высокой степенью универсальности для охвата самого широкого спектра потребителей. Разработка ПО под заказ не предусматривается, а производится выбор оптимального варианта среди конкурирующих между собой продуктов

Вертикальный/отраслевой рынок представлен продукцией конкретного сегмента рынка, охватывающего организации и предприятия определенного профиля деятельности. Рынок структурируется, как правило, на основе принятых в статистической отчетности групп отраслей экономики согласно Общероссийского классификатора видов экономической деятельности (ОКВЭД). ПП для вертикальных рынков являются достаточно специализированными, чтобы максимально соответствовать требованиям компаний выбранной отрасли или подотрасли, и разрабатываются, как правило, под заказ



Бизнес-модели разработки ПО



При реализации программных проектов компании выбирают одну из двух бизнес-моделей деятельности: разработку и продвижение собственных программных продуктов (продуктовая модель) или разработку уникального ПО «под заказ» (заказная модель)

Продуктовая бизнес-модель и ее особенности

- ▶ Использование продуктовой модели основано на способности компании поставить на рынок востребованный продукт и обеспечить его тиражирование и поддержку
- ▶ Малыми ресурсами могут быть созданы инновационные продукты, имеющие большой экономический и коммерческий потенциал
- ▶ Модель более перспективна с точки зрения оценки бизнеса компании-разработчика так как сама компания является непосредственным производителем новых проектов и технологий
- ▶ Модель обеспечивает лучшие условия для получения инвестиций в случае капитализации компании
- ▶ Модель мотивирует компанию-разработчика к пересмотру организационных процессов своей деятельности, а именно к переходу от управления программным проектом к управлению программным продуктом как объектом экономических отношений на рынке



Сравнительный анализ вариантов организации «заказного» программирования

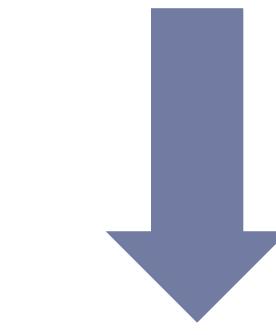
Вариант	Характеристика варианта	
	Достоинства	Недостатки
1. Реализация проекта полностью собственными силами	<ul style="list-style-type: none">1. Знание бизнес-процессов организации2. Меньшие финансовые затраты3. Независимость на этапе эксплуатации	<ul style="list-style-type: none">1. Необходимость наличия достаточно большого количества разработчиков с довольно высоким уровнем квалификации и знанием программного продукта2. Потребность в разработке методологии управления проектом и строгом ее выполнении3. Необходимость решения вопроса дальнейшей занятости сотрудников, выделенных (или нанятых) для реализации проекта
2. Реализация проекта (или его этапов) «под ключ» силами внешней компании	<ul style="list-style-type: none">1. Разработанная и обкатанная методология внедрения2. Опыт внедрения системы на нескольких предприятиях3. «Новый взгляд» на задачи предприятия.4. Способность оказания услуг в области оптимизации системы управления, владение современными методами построения систем управления5. Знание программного продукта6. Штат опытных программистов	<ul style="list-style-type: none">1. Большие финансовые затраты2. Сторонние специалисты не знают особенностей конкретного предприятия, и им требуется время на их изучение3. Проблема поддержки системы на этапе эксплуатации



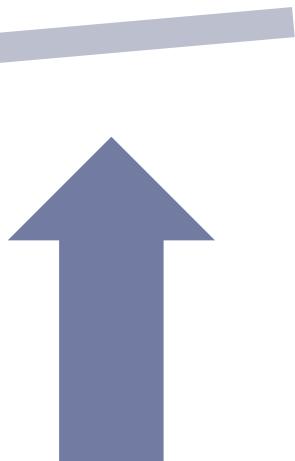
Экономика создания сложных программных продуктов

- ▶ Массовое создание сложных и дорогих программных продуктов промышленными методами и большими коллективами специалистов вызвало необходимость их достоверного экономического прогнозирования и анализа, четкой организации производства, планирования работ по затратам, этапам и срокам реализации
- ▶ Существует потребность в оценивании конкретных факторов, влияющих на экономические характеристики программных проектов, вследствие реально существующих и потенциально возможных ограничений ресурсов и необходимости экономного использования быстро возрастающих капиталовложений в производство сложных комплексов программ различного назначения, характеризующихся высокими требованиями к их качеству

Существующие противоречия при организации процесса производства ПО:

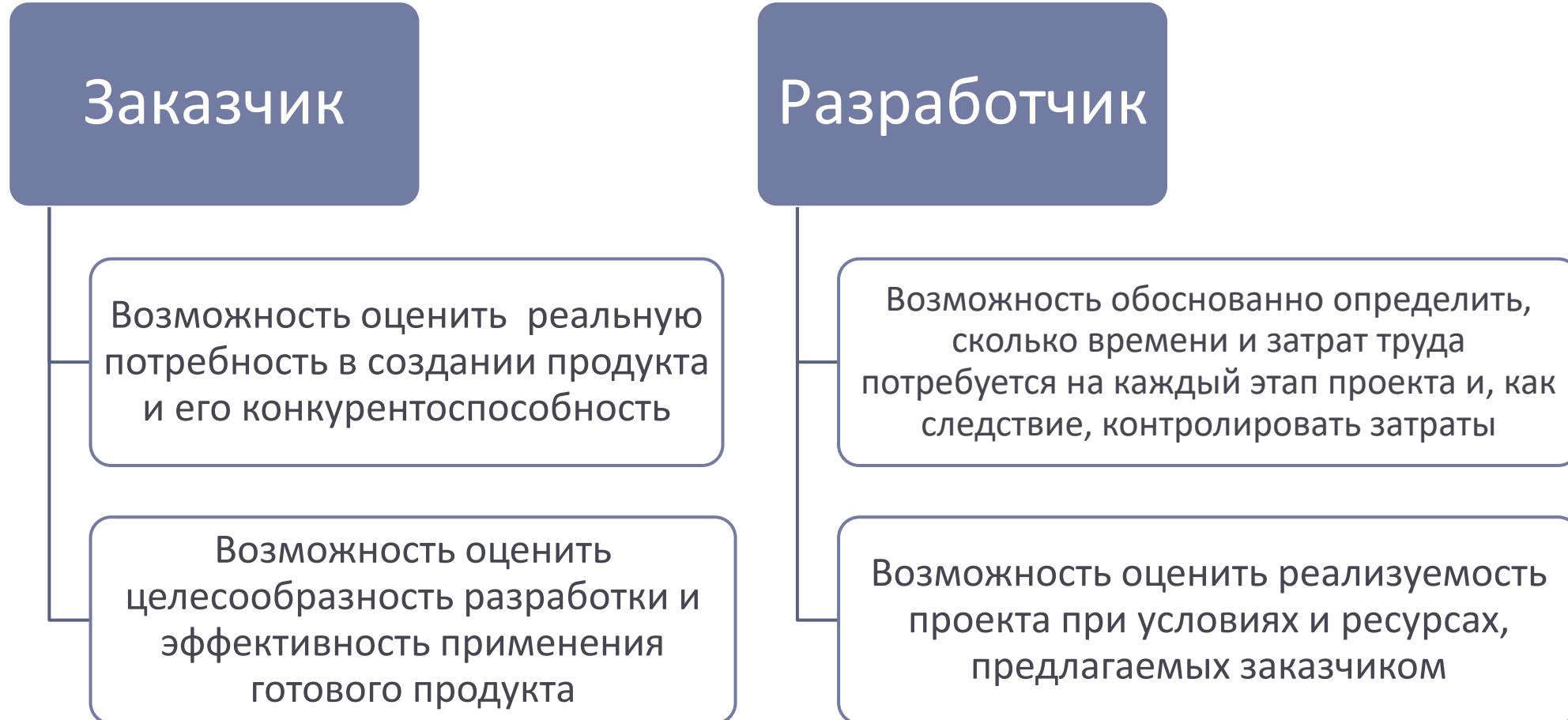


Менеджеры и разработчики программного обеспечения, как правило, не знают даже основ экономики промышленного производства сложной технической продукции



Экономисты не представляют сущность и свойства объектов разработки программных продуктов, а также особенностей технологических процессов их производства и применения

Важность понимания экономических вопросов при реализации сложных технических проектов



Экономический анализ жизненного цикла сложных программных продуктов

Анализ факторов, влияющих на экономику производства конкретных программных продуктов

- целесообразно ли проводить работы над конкретным проектом или следует его прекратить вследствие недостаточности ресурсов специалистов, времени или возможной трудоемкости производства
- при наличии достаточных ресурсов следует ли провести маркетинговые исследования для определения рентабельности проекта с целью поставки программного продукта заказчику или на рынок

Прогнозирование экономических характеристик программных продуктов

- достаточно ли полно и корректно formalизованы требования к проекту, на основе которых проводились оценки экономических характеристик, или их следует откорректировать и выполнить повторный анализ с уточненными исходными данными
- есть ли возможность применить готовые повторно используемые компоненты, рентабельно ли их применять в конкретном проекте или весь проект целесообразно разрабатывать как полностью новый

Способы сокращения погрешности оценивания экономических характеристик на начальных этапах разработки

При использовании различных методик оценивания и прогнозирования экономических характеристик, а также при применении формальных методов управления проектами необходимо учитывать следующие факторы:

- ▶ исходные тексты программных компонентов различны по размерам и сложности, и по отдельности не определяют сложность и размер конечного программного продукта
- ▶ разработка сложных продуктов требует творчества и сотрудничества разных специалистов, индивидуальное и групповое поведение которых, как правило, трудно предсказать
- ▶ в области экономики жизненного цикла сложных ПП накоплен относительно небольшой опыт анализа и количественных оценок, и его трудно увеличивать, не обобщая опыт реализованных проектов



Особенности современного производства программного обеспечения

- ▶ Принятие решений на основе анализа альтернатив с учетом максимального соответствия решаемой задаче на основе сопоставления затрат и ожидаемой прибыли
- ▶ Применение управляемого процесса разработки ПО, основанного на использовании измеримых количественных характеристик и эффективной организации командной работы
- ▶ Выполнение широкого спектра задач, начиная с исследований, разработки, проектирования, производства, тестирования, внедрения, эксплуатации и управления, и заканчивая продажами, консультированием и обучением
- ▶ Широкое использование инструментальных средств
- ▶ Ориентация на повторное использование (reuse) результатов проектирования и проектных артефактов



Технологизация производства программного обеспечения

Технологизация производства позволяет:

- ▶ автоматизировать нетворческие, технические и рутинные операции и этапы
- ▶ облегчать творческие процессы за счет отбора, обработки и отображения информации, необходимой для принятия творческих решений

Вывод: Даже при сокращении суммарных затрат на разработку программных компонентов за счет автоматизации нетворческого труда, все более определяющей для экономических характеристик создания программных продуктов становится доля затрат на творческий труд и возрастают требования к творческим способностям при отборе и обучении специалистов

На каждом этапе должен проводиться поиск эффективных технических и экономических решений реализации проекта, исследование и сопоставление альтернативных решений, которые должны приводить к достижению поставленных целей производства программного продукта



Структура трудозатрат при производстве сложных программных продуктов

- ▶ поиск альтернативных решений, новых методов и способов реализации заданных требований
- ▶ формирование и декомпозиция этих требований

Вывод: основную долю трудозатрат составляет творчество специалистов – разработчиков ПО, причем в перспективе, несмотря на автоматизацию и повышение инструментальной оснащенности производства, доля творческого труда при создании полностью новых сложных программных продуктов будет возрастать

По мере повышения квалификации коллективов и автоматизации творческой части труда следует ожидать асимптотического приближения проектов к предельным значениям относительных экономических характеристик новых разработок. Эти значения определяются интеллектуальными возможностями человека по интенсивности принятия творческих решений. Им соответствуют наличие предельных значений производительности труда и длительности разработки сложных комплексов программ



Список литературы

- ▶ Липаев В.В. Экономика производства программных продуктов. – М.: из-во «СИНТЕГ», 2011
- ▶ Ехлаков Ю.П. Управление программными проектами. – Томск.: Из-во «ТУСУР», 2015
- ▶ Ехлаков Ю.П. , Бараксанов Д.Н. , Янченко Е.А. Модели управления жизненным циклом программного продукта. – Томск.: Из-во «ТУСУР», 2013
- ▶ Орлов С.А. Технологии разработки программного обеспечения. – СПб.: из-во «Питер», 2002
- ▶ Архипенков С. Лекции по управлению программными проектами. – М., 2009
- ▶ Фредерик Брукс. Мифический человеко-месяц или как создаются программные системы. Пер. с англ. – СПб.: из-во «Символ», 2001
- ▶ Уокер Ройс. Управление проектами по созданию программного обеспечения. Пер. с англ. – М.: из-во «Лори», 2002
- ▶ Роберт Т. Фатрелл, Дональд Ф. Шафер, Линда И. Шафер. Управление программными проектами: достижение оптимального качества при минимуме затрат. Пер. с англ. - М.: Издательский дом «Вильямс», 2003



Список литературы

- ▶ Кент Бек. Экстремальное программирование. Пер. с англ. – СПб.: из-во «Питер», 2002
- ▶ Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. Пер. с англ. – СПб.: из-во «Питер», 2002
- ▶ Стив Макконнелл. Остаться в живых! Руководство для менеджера программных проектов. Пер. с англ. – СПб.: из-во «Питер», 2006
- ▶ Фергус О'Коннэл. Как успешно руководить проектами. Серебряная пуля. Пер. с англ. – М.: из-во «Кудиц-образ», 2003
- ▶ Дж. Филипс. Менеджмент ИТ-проектов: на пути от старта до финиша. Пер. с англ. – М.: из-во «Лори», 2008
- ▶ Стив Макконнелл. Сколько стоит программный проект. Пер. с англ. – СПб.: из-во «Питер», 2007
- ▶ Эдвард Йордон. Путь камикадзе: как разработчику программного обеспечения выжить в безнадежном проекте. Пер. с англ. - М.: из-во «Лори», 2001
- ▶ Оценка и аттестация зрелости процессов создания и сопровождения программных средств и систем (ISO/IEC TR 15504 – СММ). Пер. с англ. – М.: Книга и бизнес, 2001
- ▶ Рейнвотер Дж. Как пасти котов. Наставления для программистов, руководящих другими программистами. – СПб.: Питер, 2006



Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана

baryshnikovam@mail.ru

Лекция 2

Проект как способ разработки программного обеспечения.
Характеристики проекта как объекта управления. Функции
управления проектами. Список компетенций менеджера
программного проекта

Единственный способ разработать
программное обеспечение – это
инициировать проект по его созданию



Вместо введения

- ▶ По оценкам аналитического агентства The Standish Group, в 2013 году во всем мире на проекты разработки и внедрения программных продуктов было потрачено \$750 млрд. и эти затраты составили примерно 1% от мирового валового продукта
- ▶ На Соединенные Штаты приходилось около \$300 млрд. (40%), на Европу – \$200 млрд. (27%), на Азию – \$100 млрд. (13%), на остальную часть мира – \$150 млрд. (20%)



Результаты реализации проектов по разработке и внедрению ПО

	2004	2006	2008	2010	2012	2013
успешные	29%	35%	32%	37%	39%	36%
провальные	18%	19%	24%	21%	18%	16%
спорные	53%	46%	44%	42%	43%	48%

Успешные проекты – все цели проекта достигнуты в плановый срок и бюджет

Провальные проекты – проекты, остановленные без получения результата

Спорные проекты – те проекты, которые были завершены либо с превышением сроков, либо стоили дороже, чем планировалось, либо достигли лишь части целей

По оценкам The Standish Group, на так называемые «провальные» проекты потрачено \$120 млрд. Что касается «спорных» софтверных проектов, то перерасход бюджета по ним оценивается примерно в \$80 млрд. Т.е. \$200 млрд. (!), потраченных не то чтобы напрасно, но без видимой экономической выгоды



Основные выводы

- ▶ Проекты в среде высоких технологий, к которым относится и разработка программного обеспечения, очень трудно планировать и выполнять
- ▶ Еще в 70-годах Фредерик Брукс сделал утверждение, что несмотря на все достижения в области создания аппаратных и программных средств, не было никакого соответствующего развития ни в технологии, ни в методике управления, которое позволило бы выполнять программные проекты вовремя, в пределах согласованного бюджета и требований к качеству конечного продукта
- ▶ Статьи и книги, посвященные созданию программного обеспечения, уделяли повышенное внимание различным аспектам разработки (методологиям управления жизненным циклом программных проектов) и практически не касались вопросов общей дисциплины управления проектами – проектного менеджмента

«Мы искали развития. Поскольку компьютер и программное обеспечение были новы, мы думали тогда, что нужна некая *новая* вещь, чтобы работать с ними. По какой-то причине мы никак не могли себе представить, что могли бы сработать *старые* вещи».

Фергус О'Коннэл, «Серебряная пуля»



Проект

- Проект от лат. *projectus*

Проект (от лат. *projectus*) — замысел, идея, образ, намерение, обоснования, план

- **Проект – определение по ГОСТ Р 54869-2011**

Комплекс взаимосвязанных мероприятий, направленный на создание уникального продукта или услуги в условиях временных и ресурсных ограничений

- **РМВоК:**

Проект – это временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов

По данным Российской ассоциации Управления Проектами «СОВНЕТ», около 40% целенаправленной общественно полезной деятельности, реализуется через различные проекты и программы и 25% мирового бюджета тратится на проекты. Профессиональное управление проектами позволяет эффективно распределить ответственность и обязанности между участниками проекта, сэкономить до 30% времени и до 20% средств, снизить риски недостижения запланированных результатов



Интересные наблюдения

- ▶ Как только разработка программного обеспечения перестала быть уделом одиничек, а превратилась в один из видов коллективной работы, направленной на достижение определенной цели в условиях ограничений по срокам, объемам финансирования или численности команды разработчиков, по отношению к ней все чаще стали звучать определения «программный проект», «проект создания ПО», «проект разработки»
- ▶ Чем сложнее бизнес-задача, чем важнее программный проект, направленный на ее решение, тем чаще приходится оперировать общими управленческими понятиями и применять знания и навыки, характерные для общего менеджмента, в том числе – проектного, т.е. «управления проектами»
- ▶ Проведенные исследования ничего не могут сказать о влиянии на успешность проектов факта применения формальных методологий разработки ПО, зато они четко показывают влияние уровня компетенции менеджеров проектов на результат. Успешное завершение и, тем более, спасение тонущего проекта тем вероятнее, чем большим кругозором обладает лидер проекта, пришел ли он в ИТ как «чистый управленец» или вырос из среды разработчиков



Определение проекта

- ▶ Проект- это что-либо, что задумывается или планируется, большое предприятие
- ▶ Проект-это временное предприятие, предназначенное для создания продукта или услуги
- ▶ Проект- это отдельное предприятие с определенными целями, часто включающими требования по времени, стоимости, качеству

Проект — произвольный ряд действий или задач, имеющих определенную цель, которая будет достигнута в рамках выполнения некоторых заданий, характеризующихся определенными датами начала и окончания, пределами финансирования и ресурсами



Признаки проекта

- ▶ направленность проекта на достижение конкретной цели, выраженной в требованиях к конечному результату, определяемому в терминах требуемых ресурсов, качества и времени реализации
- ▶ уникальность проекта как разового (неповторяющегося) мероприятия, требующего специфической организации управления
- ▶ ограниченность проекта по времени и ресурсам (финансовым, трудовым, материальным) и, как следствие, необходимость нахождения постоянного компромисса между объемом работ, временем, качеством и рисками и их перераспределения в ходе выполнения проекта
- ▶ структурная сложность проекта как комплекса тесно увязанных мероприятий и его высокая неопределенность, обусловленная возможными изменениями условий реализации, потребности в тех или иных видах ресурсов



ПРОЕКТ – это ограниченное по времени, целенаправленное изменение отдельной системы с установленными требованиями к качеству результатов, возможными рамками расхода средств и ресурсов и специфической организацией



Классификация проектов

По классу:

- ▶ монопроект
- ▶ мультипроект
- ▶ мегапроект

По виду:

- ▶ учебно-образовательные
- ▶ проекты исследования и развития
- ▶ инновационные
- ▶ инвестиционные
- ▶ комбинированные

По типу:

- ▶ социальные
- ▶ экономические
- ▶ организационные
- ▶ технические
- ▶ смешанные

По длительности:

- ▶ краткосрочные (до 3 лет)
- ▶ среднесрочные (от 3 до 5 лет)
- ▶ долгосрочные (свыше 5 лет)



Жизненный цикл проекта

Фаза концепции - формулирование результатов проекта, сбор исходных данных и предварительное обследование, разработка альтернативных вариантов реализации и сравнительная оценка альтернатив, выработка концепции реализации проекта, ее апробация и экспертиза



Фаза планирования - разработка плана проекта, назначение руководителя, формирование команды, заключение договоров с поставщиками и подрядчиками

Фаза реализации - выполнение работ проекта, их координация и материально-техническое обеспечение, оперативный контроль и регулирование основных показателей проекта, мотивация и стимулирование команды проекта



Фаза завершения - эксплуатационные испытания конечного продукта, подготовка кадров для его эксплуатации, подведение итогов, оформление документации и закрытие проекта



Участники проекта

- ▶ Инициатор
- ▶ Владелец (иногда это клиент или заказчик)
- ▶ Инвесторы
- ▶ Исполнитель (команда проекта)
- ▶ Поставщики, подрядчики, потребители
- ▶ Менеджер
- ▶ Другие участники (спонсоры, консультанты, конкуренты, лицензиары, общественные группы и население)



Окружение проекта

- ▶ проект возникает, существует и развивается в определенном окружении, называемом внешней средой
- ▶ проект, как и всякая система, может быть разделен на элементы, при этом между выделяемыми элементами должны определяться и поддерживаться определенные связи
- ▶ состав проекта не остается неизменным в процессе его реализации и развития: в него могут включаться новые элементы, из него могут удаляться некоторые элементы

Важную роль при выборе решений по реализации программного проекта играют ограничения и допущения

Например, **ограничения** могут содержать следующие требования: обязательную сертификацию продукта; использование конкретной заданной программно-аппаратной платформы; специфические требования к защите информации

Допущения — события или действия, принимаемые как данность. Например, оценивая проект по схеме с фиксированной ценой, можно записать в допущении предположение о том, что стоимость лицензий на программное обеспечение (ПО), приобретаемое на стороне, не изменится до завершения проекта. Допущения проекта должны быть оформлены документально и заранее доведены до сведения заказчика

Характеристики проекта как объекта управления

- ▶ **Назначение проекта** - новые продукты или услуги, которые получит потребитель в результате реализации проекта
- ▶ **Стоимость проекта** - сметные затраты, необходимые для выполнения работ проекта
- ▶ **Объемы работ проекта** - количественные показатели работ
- ▶ **Сроки выполнения проекта** - даты начала, окончания, продолжительность
- ▶ **Качество проекта** - соответствие характеристик проекта и его результатов установленным стандартам качества
- ▶ **Ресурсы проекта** – совокупность ресурсов, требующихся для осуществления проекта: оборудование, материалы, персонал, программное обеспечение, производственные площади и т.д.
- ▶ **Исполнители проекта** - специалисты и организации, привлеченные к выполнению работ проекта, их количество, состав и квалификация
- ▶ **Риски проекта** - определение рискованных событий в проекте, вероятности их свершения и ущерба от их воздействия на проект



Основные аспекты, влияющие на целесообразность инициации проекта

- ▶ технические (используются ли в нем лучшие из технических альтернатив)
- ▶ коммерческие (является ли проект перспективным с точки зрения спроса на продукцию проекта)
- ▶ финансовые (возместятся ли затраты на реализацию проекта, какова его рентабельность)
- ▶ экологические (какое влияние оказывает проект на окружающую среду)
- ▶ организационные (имеется ли ответственная в целом за проект организация и как она выполняет возложенные на нее функции)
- ▶ социальные (отражает ли проект местные условия, совместим ли он с обычаями, традициями заинтересованных участников)



Управление проектами

методология организации, планирования, руководства, координации трудовых, финансовых и материально-технических ресурсов на протяжении проектного цикла, направленная на эффективное достижение его целей путем применения современных методов, техники и технологии управления для достижения определенных в проекте результатов по составу и объему работ, стоимости, времени, качеству и удовлетворению участников проекта

В основе методов управления проектами лежат методики сетевого планирования, разработанные в конце 50-х годов в США. В 1956 г. М. Уолкер из фирмы «Дюпон», исследуя возможности более эффективного использования принадлежащей фирме вычислительной машины Univac, объединил свои усилия с Д. Келли из группы планирования капитального строительства фирмы «Ремингтон Рэнд». Они попытались использовать ЭВМ для составления планов-графиков крупных комплексов работ по модернизации заводов фирмы «Дюпон». В результате был создан рациональный и простой метод описания проекта с использованием ЭВМ. Первоначально он был назван методом Уолкера-Келли, а позже получил название *Метода Критического Пути* - МКП (или CPM – Critical Path Method)



Методы управления проектами позволяют:

- ▶ определить цели проекта и провести его обоснование;
 - ▶ выявить структуру проекта (подцели, основные этапы, задачи, которые предстоит выполнить)
 - ▶ определить необходимые объемы и источники финансирования
 - ▶ подобрать исполнителей, в частности через процедуры торгов и конкурсов
 - ▶ подготовить и заключить контракты
 - ▶ определить сроки выполнения проекта, составить график его реализации, рассчитать необходимые ресурсы
 - ▶ обосновать смету и бюджет проекта
 - ▶ планировать и учитывать риски
 - ▶ обеспечить контроль за ходом выполнения проекта
-

Чем управляет управление проектами?

- ▶ предметной областью проекта, т.е. содержанием проекта
- ▶ качеством
- ▶ временем
- ▶ стоимостью
- ▶ рисками
- ▶ ресурсами
- ▶ коммуникациями
- ▶ поставками и контрактами
- ▶ изменениями



Управление предметной областью

Предметная область проекта определяется целями, результатами и составом работ. Фактически под управлением содержанием понимается формальное принятие решения о начале проекта, планирование объема работ, декомпозиция всего объема работ на мелкие измеримые задачи, формальная проверка приемлемости результатов работ, изменение объема работ

Следует учитывать, что в процессе реализации проекта все составляющие предметной области претерпевают изменения:

- ▶ цели, результаты и состав работ могут изменяться или уточняться как в процессе разработки проекта, так и по мере достижения промежуточных результатов
- ▶ объемы работ могут уточняться в процессе реализации проекта и по мере его продвижения изменяются от нуля до 100% при завершении работ

Управление содержанием заключается в работе с этими изменениями на протяжении всего жизненного цикла проекта



Управление временем

Время является одним из определяющих факторов в оценке успеха проекта. Оно требует особого внимания, поскольку является невосполнимым ресурсом.

Функция управления временем реализуется посредством календарного планирования работ, их отслеживания, актуализации и корректировки и включает в себя:

- ▶ определение перечня работ и их продолжительности,
- ▶ установление сроков начала и завершения отдельных работ, этапов и проекта в целом,
- ▶ определение времени наступления важнейших (контрольных) событий,
- ▶ минимизацию (оптимизацию) временных характеристик проекта,
- ▶ разумное использование резервов времени,
- ▶ контроль за развитием проекта по его временным характеристикам,
- ▶ прогнозирование сроков завершения работ,
- ▶ принятие решений по ликвидации нежелательных временных отклонений



Управление качеством

Реализуется через установление требований и стандартов как к качеству результатов проекта, так и к качеству процесса его реализации. Качество обеспечивается через проектные, организационные и управленческие решения, используемые материалы и оборудование и пр.

Управление стоимостью

Включает планирование ресурсов, предварительную оценку расходов, связанных с проектом, и подготовку сметы, контроль расходования и поступления денежных средств и принятие решений в случае превышения расходов и других отклонений от финансовых планов. Главной задачей управления стоимостью является соблюдение бюджетных рамок проекта, и получение предусмотренной прибыли от его осуществления



Управление персоналом

Управление персоналом включает в себя:

- ▶ идентификацию, документирование и назначение проектных ролей участникам проекта и структуры их отчетности
- ▶ подбор и получение необходимых для проекта трудовых ресурсов
- ▶ распределение персонала по работам проекта
- ▶ формирование планов повышения квалификации исполнителей проекта
- ▶ стимулирование индивидуальной и командной производительности труда



Управление рисками

Риск в контексте проекта рассматривается как воздействие на проект и его элементы непредвиденных событий, которые могут нанести определенный ущерб и препятствовать достижению целей проекта. Риск проекта характеризуется тремя факторами: событиями, оказывающими негативное воздействие на проект, вероятностью наступления таких событий и оценкой ущерба, нанесенного проекту такими событиями

Управление контрактами и поставками

Включает процессы выбора стратегии контрактной деятельности, информационно-рекламную работу, определение состава, номенклатуры и сроков работы привлекаемых по контракту субъектов, подготовку контрактных предложений, выбор контрагентов и поставщиков путем торгов, конкурсов, тендеров и др., подготовку документации на заключение контрактов, контроль за ходом их выполнения, закрытие и расчет по завершенным контрактам



Управление коммуникациями

Управление проектом в целом зависит от успешной организации взаимодействия всех участников проекта и обеспечения их потребности в информации, необходимой для его осуществления. Разработка, организация и контроль процесса информационного обмена для удовлетворения потребностей всех участников проекта осуществляется через управление коммуникациями. В эту функцию обычно включаются: планирование коммуникаций, распределение информации, подготовка отчетов, приемка проекта, администрирование и закрытие проекта

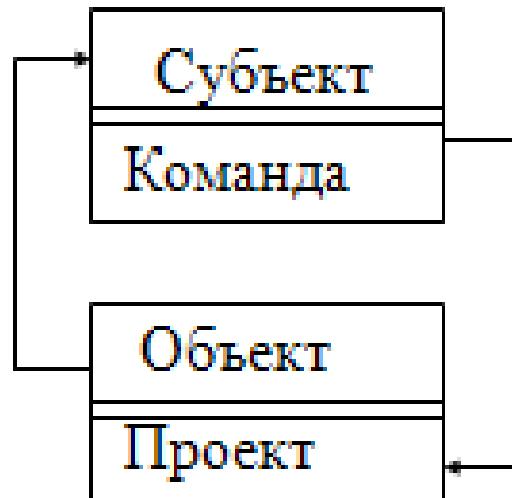
Управление изменениями

Включает в себя прогнозирование и выявление изменений, разработку плана изменений, его исполнение и оценку результатов. Управление изменениями является интегрирующим фактором в проекте и тесно связано со всеми процессами и функциями, где происходят или могут произойти изменения: предметной областью, временем, стоимостью, качеством, риском, контрактами/поставками, человеческими ресурсами, коммуникациями и др.



Кибернетическая схема управления проектом

В системе управления проектом реализуются две группы процессов:



- ▶ проектно-ориентированные процессы, которые связаны с объектом УП, т.е. с самим проектом, выполняются исполнителями работ проекта и направлены на достижение результатов проекта — создание нового продукта или услуги
- ▶ процессы управления проектом, связанные с субъектом УП, или командой управления проектом и ее деятельностью по описанию, планированию, организации и координации работ в проекте для обеспечения его успешного завершения



Виды деятельности, направленные на реализацию функций управления

- ▶ **Планирование** - это определение оптимального результата при заданных ограничениях по времени и ресурсам
- ▶ **Организация** - определение путей, методов и средств достижения поставленной цели
- ▶ **Координация** – установление взаимодействия в совместном труде участников планируемого процесса
- ▶ **Активизация или мотивация** - создание таких стимулирующих условий труда, при которых каждый работник трудился бы с наибольшей отдачей
- ▶ **Контроль** – прогнозирование отклонений для их своевременного предупреждения



10 правил промышленного создания ПО или принципы традиционного управления программными проектами

1. Поиск и обнаружение ошибки в ПО после его сдачи в эксплуатацию обходится в 100 раз дороже, чем поиск и обнаружение ошибки на ранних стадиях разработки
2. Можно сократить срок разработки ПО на 25% от номинального, но не более
3. На каждый доллар, вложенный в разработку, приходится тратить два доллара на сопровождение
4. Стоимость разработки и сопровождения программного обеспечения является, прежде всего, функцией числа строк исходного кода
5. Различия в уровне квалификации разработчиков приводят к огромной разнице в продуктивности при создании программного обеспечения
6. Общее отношение стоимости программного обеспечения к стоимости аппаратного обеспечения продолжает расти. В 1955 г. оно составляло 15:85; в 1985 г. - 85:15
7. При создании ПО всего лишь около 15% усилий затрачивается собственно на программирование
8. Программные системы и продукты обычно стоят в три раза дороже в пересчете на одну строку исходного кода, чем отдельные программы. Продукты, состоящие из программных систем (т.е. системы систем), стоят дороже в девять раз
9. Сквозной контроль позволяет обнаружить 60% ошибок
10. 80% работы выполняют 20% работающих



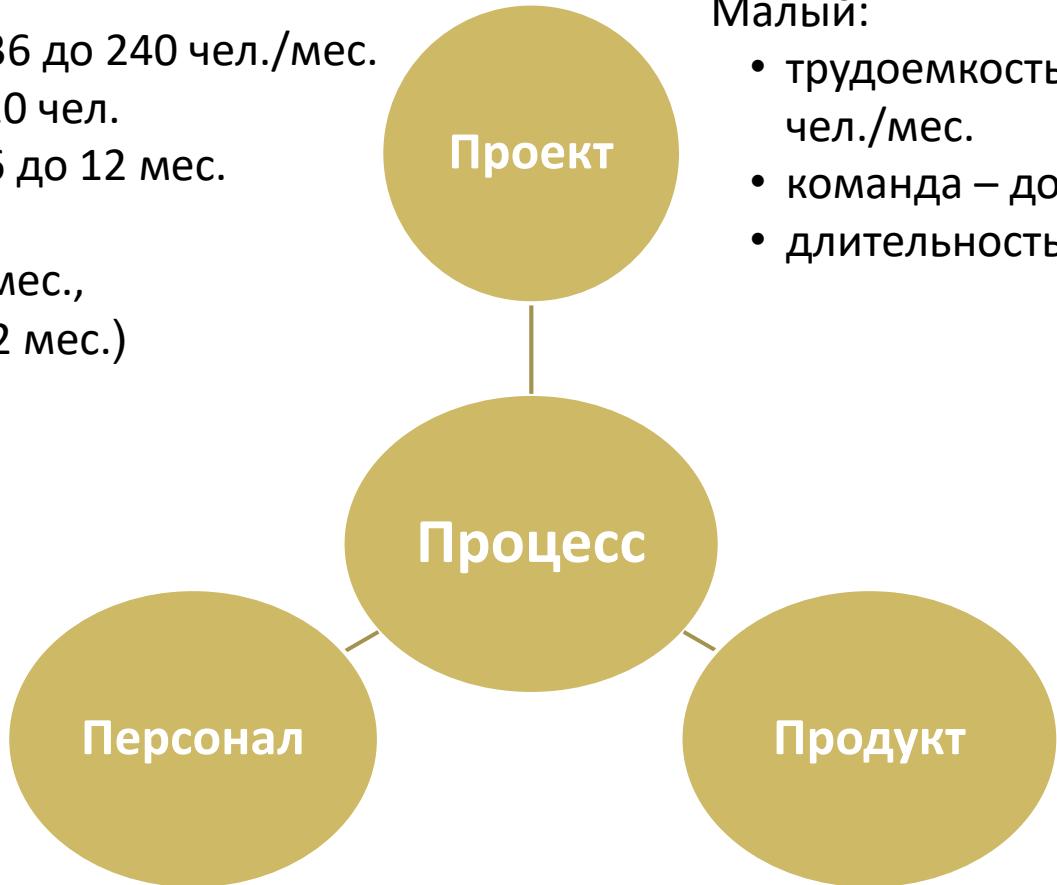
Четыре «П» — в разработке программного обеспечения

Средний:

- трудоемкость – от 36 до 240 чел./мес.
- команда – от 5 до 20 чел.
- длительность – от 6 до 12 мес.

Большой (>240 чел./мес.,
>20 чел., >12 мес.)

Профессионализм
Сработанность
Стабильность
Мотивация
Эффективность
коммуникаций



Малый:

- трудоемкость – от 6 до 36 чел./мес.
- команда – до 5 чел.
- длительность – до 6 мес.

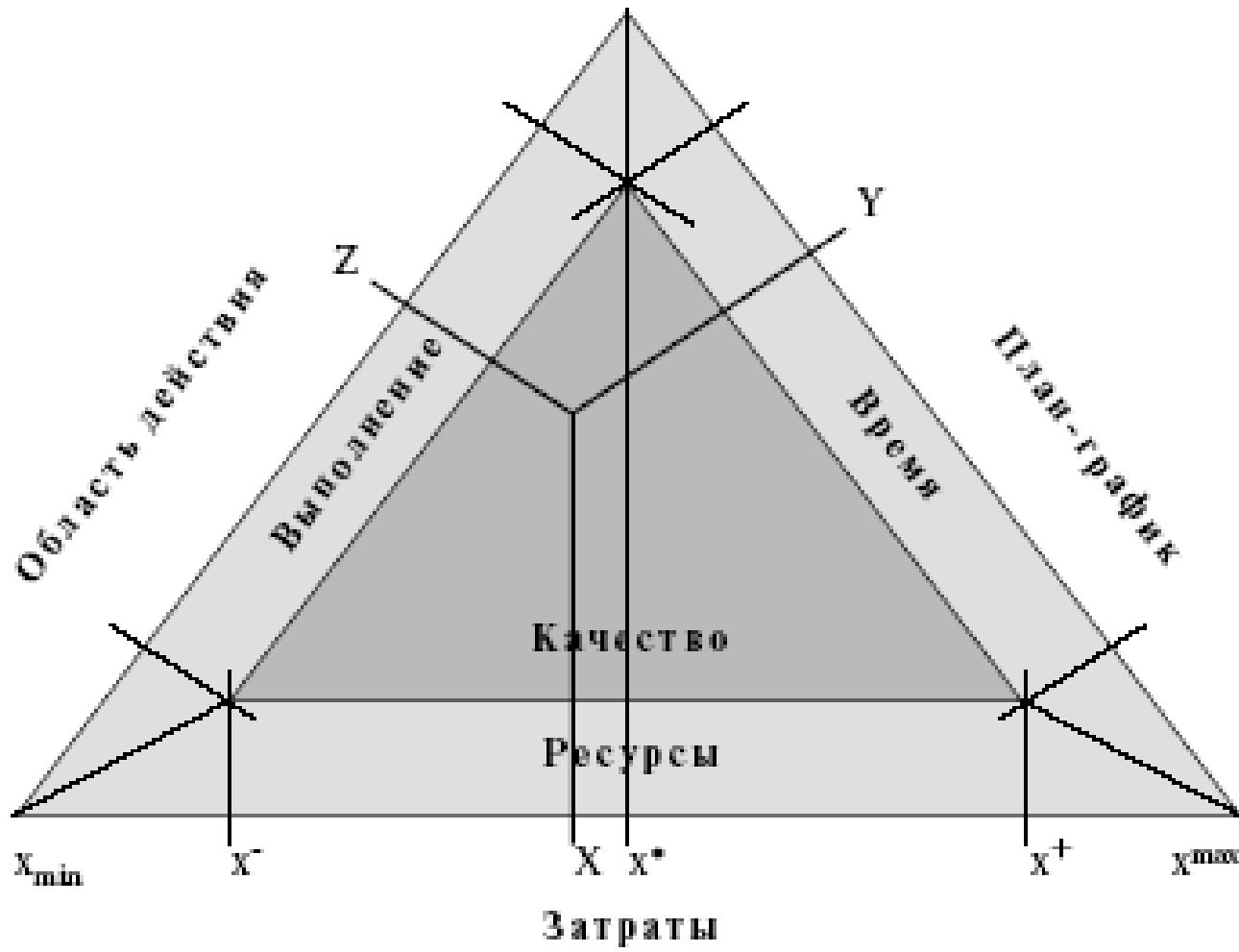
Техническая сложность:

- новый продукт
- новые технологии

Критичность для
заказчика



Треугольник менеджмента проектов



Менеджмент проекта можно рассматривать как обеспечение поставок продукта, разработка которого требует выполнения определенного объема работ с привлечением затрат, не выходящих за определенные пределы, укладывающееся в заданные временные рамки и удовлетворяющее приемлемому уровню качества. По разным причинам соблюдать баланс по всем параметрам чаще всего не удается, а потому приходится выбирать в качестве первичной цели только один или два параметра, ставя остальные в зависимость от них



Список компетенций менеджера программного проекта

Продукт	Проект	Персонал
<ul style="list-style-type: none">● Определение критериев для выполнения экспертных оценок● Знание стандартов процесса● Определение продукта● Оценка альтернативных процессов● Управление требованиями● Управление субподрядчиками● Выполнение начальной оценки● Отбор методик и инструментов● Подгонка процессов● Отслеживание качества продукта● Понимание действий по разработке продукта	<ul style="list-style-type: none">● Создание структуры пооперационного выполнения работ● Документирование плана● Оценка стоимости● Оценка трудозатрат● Менеджмент рисков● Отслеживание процесса разработки● Составление графика● Выбор метрических показателей● Отбор инструментов менеджмента проектов● Отслеживание процессов● Отслеживание хода разработки продукта	<ul style="list-style-type: none">● Оценка производительности● Вопросы интеллектуальной собственности● Организация эффективных встреч● Взаимодействие и общение● Лидерство● Управление изменениями● Успешное ведение переговоров● Планирование карьерного роста● Эффективное представление● Набор персонала● Отбор команды● Создание команды



Методика разработки продукта

- ▶ Определение критериев для выполнения экспертных оценок (обзора). В обзоре описывается действие оценивания или приводится оценка конечных продуктов проекта
- ▶ Понимание стандартов процесса разработки ПО (например, PMI, IEEE, ISO, ANSI и др.)
- ▶ Идентификация клиентской среды и требований, предъявляемых к продукту
- ▶ Оценивание различных применяемых подходов к разработке ПО
- ▶ Мониторинг изменения требований
- ▶ Планирование, управление и осуществление контроля за деятельностью субподрядчиков
- ▶ Оценивание трудностей, рисков, затрат и графика
- ▶ Определение процессов отбора инструментов для автоматизированного проектирования и разработки программ (Case-средств), инструментов оценки параметров проекта (СОСОМО, SLIM и др.), инструментов для планирования, отслеживания и контроля проекта, составления графиков и отчетов
- ▶ Изменение стандартных процессов в целях удовлетворения требований проекта
- ▶ Отслеживание качества разрабатываемых продуктов и его обеспечения
- ▶ Понимание действий по разработке продукта, изучение жизненного цикла разработки ПО и его различных моделей



Навыки менеджмента проектов

- ▶ Создание структуры пооперационного выполнения работ (WBS) для проекта
- ▶ Документирование плана, идентификация ключевых компонентов
- ▶ Оценка стоимости завершения проекта
- ▶ Оценка трудозатрат, необходимых для завершения проекта
- ▶ Идентификация, определение воздействия и обработка рисков
- ▶ Отслеживание процесса разработки ПО
- ▶ Составление графика и ключевых стадий проекта
- ▶ Выбор и использование соответствующих метрических показателей (критериев) разработки ПО
- ▶ Отбор инструментов менеджмента проектов
- ▶ Мониторинг совместимости среди членов команды разработчиков проекта
- ▶ Отслеживание хода разработки продукта, мониторинг хода реализации проекта с помощью выбранных метрических показателей



Навыки менеджмента персонала

- ▶ Оценка действий команды, направленная на улучшение ее производительности
 - ▶ Понимание степени воздействия вопросов интеллектуальной собственности на ход реализации проекта (знание основных норм права, связанных с разработкой ПО)
 - ▶ Планирование и проведение эффективных встреч
 - ▶ Сотрудничество с разработчиками, высшим руководством и другими командами
 - ▶ Обучение проектных команд для получения оптимальных результатов
 - ▶ Управление изменениями, освоение роли эффективного генератора изменений
 - ▶ Разрешение конфликтов и успешное ведение переговоров
 - ▶ Структурирование и определение руководства карьерным ростом членов команды
 - ▶ Эффективное использование письменных и устных навыков
 - ▶ Набор персонала, успешное привлечение и собеседование с членами команды
 - ▶ Отбор высококомпетентных команд
 - ▶ Формирование, руководство и поддержка эффективной команды
-

Что надо делать для успеха программного проекта

- ▶ Четко ставить цели
- ▶ Определять способ достижения целей
- ▶ Контролировать и управлять реализацией
- ▶ Анализировать угрозы и противодействовать им
- ▶ Создавать команду

С. Макконнелл

«Остаться в живых. Руководство для менеджеров программных проектов»





Спасибо за внимание!

Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана

baryshnikovam@mail.ru

Лекция 3

Жизненный цикл программного обеспечения и его модели. Модели качества программного обеспечения. Модель Фазы – Функции: Модель Гантера

Успех проекта по разработке ПО зависит не только от получения удачного программного продукта, но также от получения удачных процессов его создания



Основные виды деятельности, выполняемые при создании ПО

- ▶ проектирование — выделение отдельных модулей и определение связей между ними с целью минимизации зависимостей между частями проекта
 - ▶ кодирование — разработка кода отдельных модулей
 - ▶ тестирование — проверка работоспособности программной системы



Жизненный цикл программного обеспечения – это период времени, который начинается с момента принятия решения о необходимости создания программного обеспечения и заканчивается в момент его полного изъятия из эксплуатации

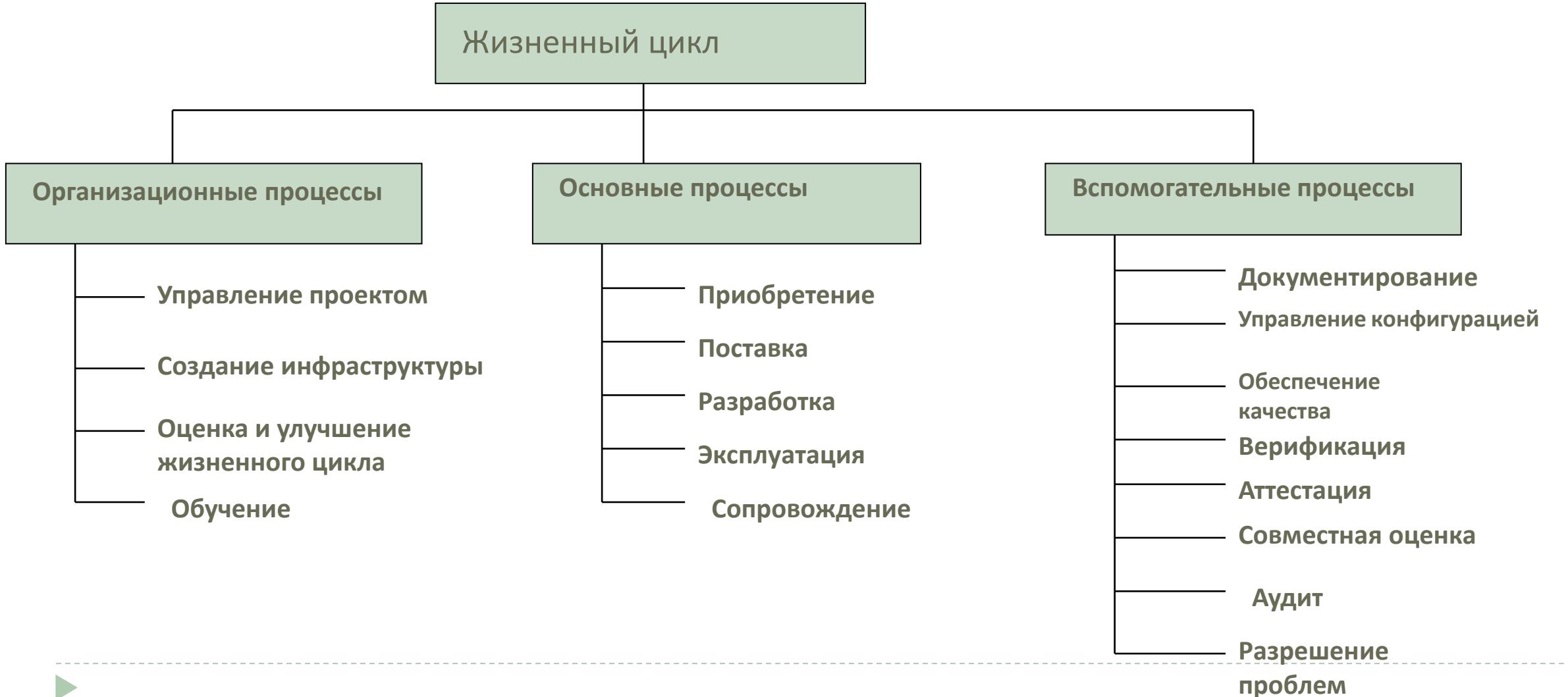
(IEEE Std. 610.12 – 1990 Standard Glossary of Software Engineering Terminology)

Основные понятия, участвующие в определении жизненного цикла

- ▶ **Артефакты** — создаваемые человеком информационные сущности – документы, в достаточно общем смысле участвующие в качестве входных данных и получающиеся в качестве результатов различных деятельности.
- ▶ **Роль** - абстрактная группа заинтересованных лиц, участвующих в деятельности по созданию и эксплуатации системы, решаяющих одни и те же задачи или имеющих одни и те же интересы по отношению к ней
- ▶ **Программный продукт** – набор компьютерных программ, процедур и, возможно связанных с ними документации и данных
- ▶ **Процесс** – совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные



Жизненный цикл ПО согласно стандарту ISO/IEC 12207: 1995 «International Technology – Software Life Cycle Processes» (ГОСТ ИСО МЭК 12207-99
Информационные технологии. Жизненный цикл программного обеспечения)



Содержание основных процессов ЖЦ ПО (ISO/IEC 12207) на примере информационной системы (ИС)

Процесс (исполнитель процесса)	Действия	Вход	Результат
Приобретение (Заказчик)	Инициирование	Решение о начале работ по внедрению ИС	Технико-экономическое обоснование внедрения ИС
	Подготовка заявочных предложений	Результаты обследования деятельности заказчика	Техническое задание на ИС
	Подготовка договора	Результаты анализа рынка / тендера	Договор на поставку / разработку
	Контроль деятельности поставщика	План поставки / разработки	Акты приемки этапов работы
	Приемка ИС	Комплексные испытания ИС	Акт приемно-сдаточных испытаний



Содержание основных процессов ЖЦ ПО (ISO/IEC 12207)

Процесс (исполнитель процесса)	Действия	Вход	Результат
Поставка (Разработчик)	Инициирование	Техническое задание на ИС	Решение об участии в разработке
	Ответ на заявочные предложения	Решение руководства об участии в разработке	Коммерческие предложения / конкурсная заявка
	Подготовка договора	Результаты тендера	Договор на поставку / разработку
	Планирование исполнения	План управления проектом	Реализация / корректировка
	Поставка ИС	Разработанная ИС и документация к ней	Акт приемно-сдаточных испытаний



Содержание основных процессов ЖЦ ПО ИС (ISO/IEC 12207)

Процесс (исполнитель процесса)	Действия	Вход	Результат
Разработка (Разработчик)	Подготовка	Техническое задание на ИС	Используемая модель ЖЦ, стандарты разработки
	Анализ требований к ИС	Техническое задание на ИС, модель ЖЦ	План работ
	Проектирование архитектуры ИС	Техническое задание на ИС	Состав подсистем, компоненты оборудования
	Разработка требований к ПО	Подсистемы ИС	Спецификации требований к компонентам ПО
	Проектирование архитектуры ПО	Спецификации требований к компонентам ПО	Состав компонентов ПО, интерфейсы с БД, план интеграции ПО



Содержание основных процессов ЖЦ ПО ИС (ISO/IEC 12207)

Процесс (исполнитель процесса)	Действия	Вход	Результат
Разработка (Разработчик)	Детальное проектирование ПО	Архитектура ПО	Проект БД, спецификации интерфейсов между компонентами ПО, требования к тестам
	Кодирование и тестирование ПО	Материалы детального проектирования	Тексты модулей ПО, акты автономного тестирования
	Интеграция ПО и квалификационное тестирование ПО	План интеграции ПО, тесты	Оценка соответствия ПО требованиям ТЗ
	Интеграция ИС и квалификационное тестирование ИС	Архитектура ИС, ПО, документация на ИС, тесты	Оценка соответствия ПО, БД, технического комплекса и комплекта документации требованиям ТЗ



Назначение моделей жизненного цикла ПО

- ▶ Модель жизненного цикла дает рекомендации по организации процесса разработки ПО в целом, конкретизируя его до видов деятельности, артефактов, ролей и их взаимосвязей
- ▶ Модель жизненного цикла служит основой для планирования программного проекта, позволяет экономнее расходовать ресурсы и добиваться более высокого качества управления
- ▶ Знание технологических функций, которые на разных этапах должны выполнять разработчики, занимающие те или иные роли, способствует правильному распределению обязанностей сотрудников

Знание моделей жизненного цикла помогает понять даже непрофессионалу, на что можно рассчитывать при заказе или приобретении программного обеспечения, а что нереально требовать от него



Модель жизненного цикла ПО

Модель жизненного цикла программного обеспечения представляет собой совокупность упорядоченных во времени, взаимосвязанных и объединенных в стадии работ, выполнение которых необходимо и достаточно для создания ПО, соответствующего заданным требованиям

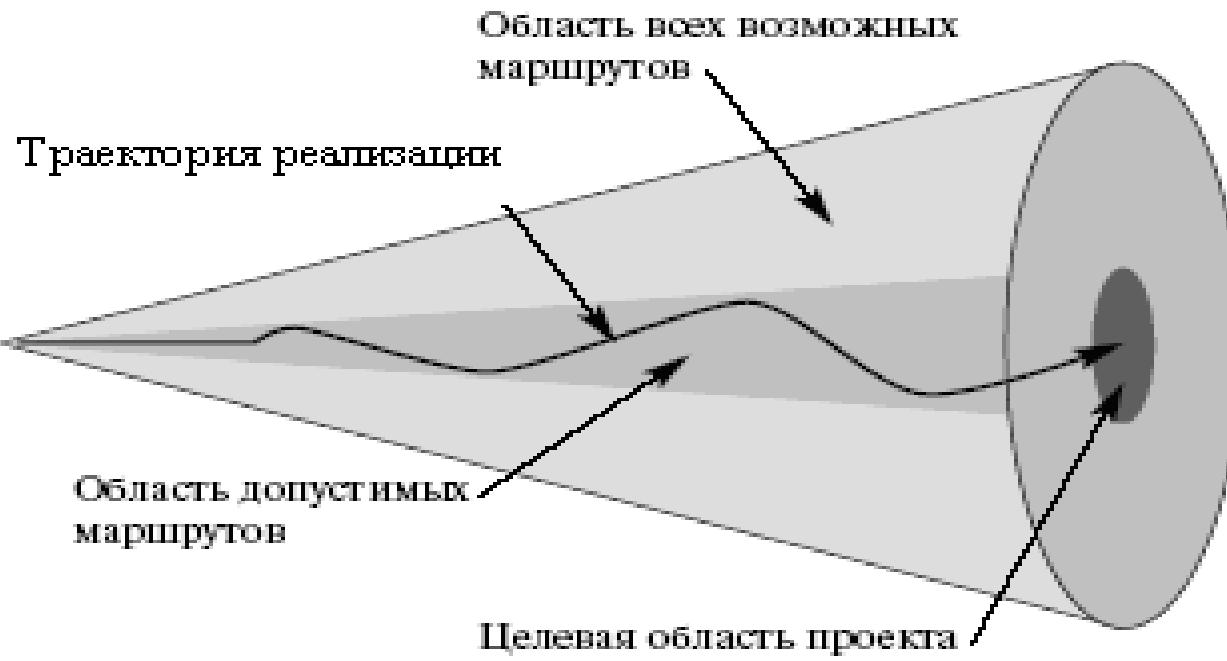
Модель жизненного цикла зависит от специфики, масштаба и сложности проекта, а также от условий, в которых система создается и функционирует. Каждая модель ЖЦ увязывается с конкретными методиками разработки программных систем и соответствующими стандартами в области программной инженерии

Организационные аспекты, подлежащие рассмотрению при формировании ЖЦ ПО:

- ▶ планирование последовательности работ и сроков их исполнения
- ▶ подбор и подготовка ресурсов (людских, программных и технических) для выполнения работ
- ▶ оценка возможностей реализации проекта в заданные сроки, в пределах указанной стоимости и др.



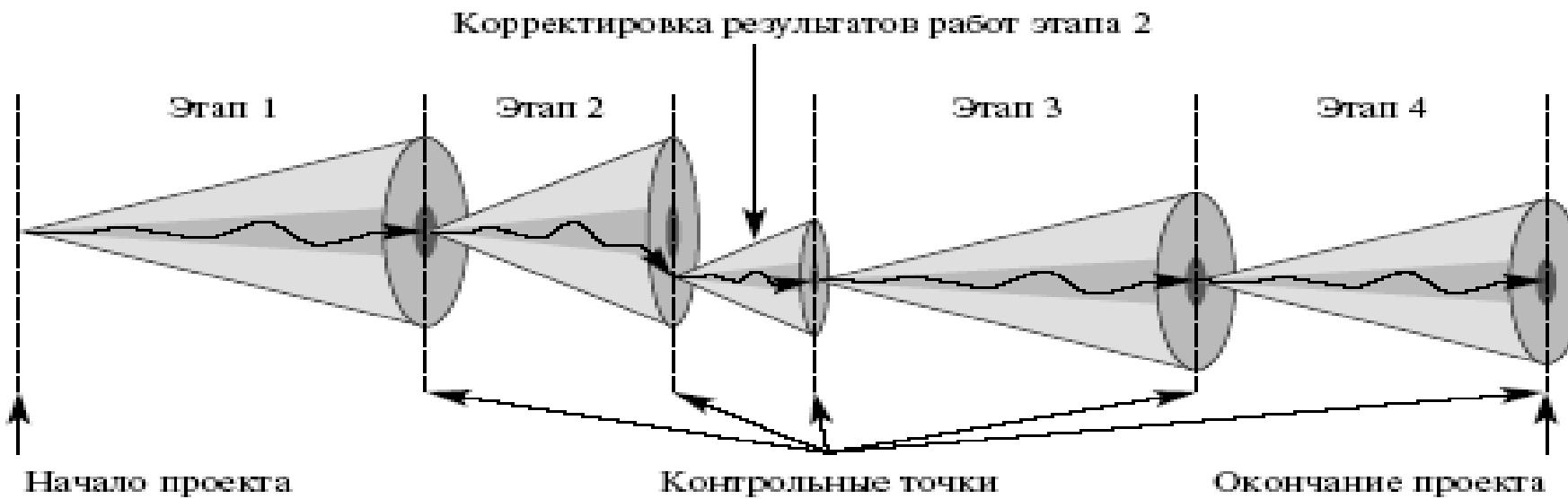
Конус операционных маршрутов проекта по разработке ПО



Одна из основных задач менеджмента программного проекта заключается в том, чтобы не допустить отклонения траекторий деятельности исполнителей от целевой области проекта. Для этого должны быть предусмотрены средства, позволяющие выявлять отклонения, и инструменты воздействия, предназначенные для их корректировки



Последовательное развитие проекта



Процесс разработки программного обеспечения разбивается на этапы. Каждый этап характеризуется:

- субъектом-исполнителем;
- сроками, когда должны быть решены задачи;
- выделенными ресурсами;
- средствами, инструментами и методами решения задач;
- контрольными мероприятиями, позволяющими удостовериться, что задачи этапа решены



Последовательные модели жизненного цикла

- ▶ Каскадная
- ▶ V-образная
- ▶ Упрощенный процесс системного проектирования

Краткое описание подхода к разработке ПО: Вначале собираются и анализируются требования. После их документирования и утверждения последовательно выполняются проектирование, кодирование и тестирование компонентов программной системы, после чего происходит их интеграция

После разработки документации система передается на сопровождение в условиях эксплуатации

Каждый этап выполняется однократно, возвратов на предыдущие этапы не предусматривается. Иногда отдельные этапы (например, связанные с верификацией) выполняются параллельно (V-образная модель)



Достоинства последовательных моделей ЖЦ

- ▶ Последовательные модели просты и удобны в использовании, так как процесс разработки выполняется поэтапно
- ▶ Они хорошо известны потребителям
- ▶ Модели хорошо срабатывают тогда, когда требования к качеству доминируют над требованиями к затратам и графику выполнения проекта
- ▶ Они способствуют осуществлению четкого менеджмента проекта, так как допускают использование инструментов временного планирования и контроля, поскольку момент завершения каждой фазы может рассматриваться в качестве контрольной точки, позволяющей проанализировать достигнутые результаты



Недостатки последовательных моделей ЖЦ

- ▶ В основе разработки программной системы лежат спецификации, вследствие чего высок риск создания системы, не удовлетворяющей потребностям пользователей, так как пользователи не в состоянии сразу изложить все свои требования и/или за время разработки могут произойти изменения во внешней среде, которые повлияют на требования к системе
- ▶ В последовательных моделях не учтены итерации между фазами и не предусмотрено внесение динамических изменений на разных этапах жизненного цикла
- ▶ Интеграция является конечной, а не пошаговой операцией
- ▶ Согласование получаемых результатов с пользователями производится только в точках, планируемых после завершения каждой стадии
- ▶ В последовательные модели не входят действия, направленные на анализ рисков

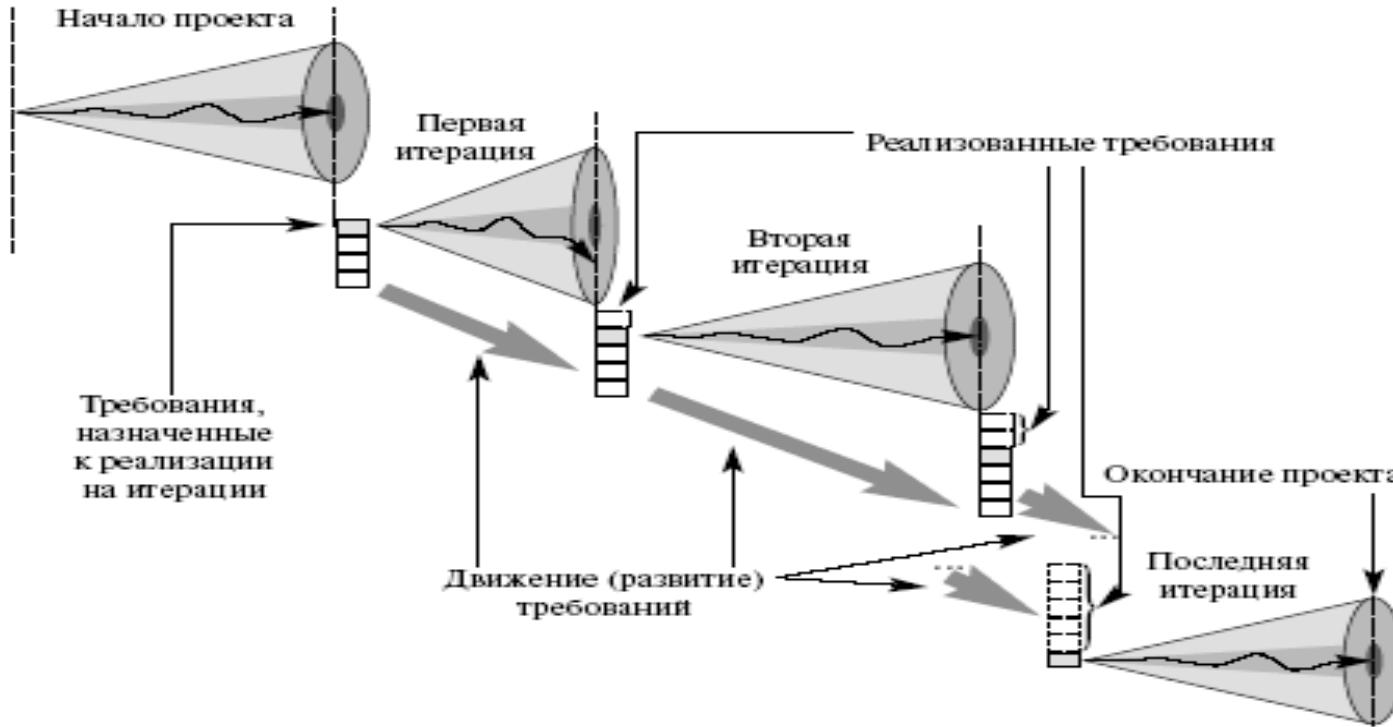


Сфера применения последовательных моделей ЖЦ

- ▶ в случаях, когда информация о требованиях известна заранее и они стабильны в течение всего периода разработки
- ▶ в проектах с участием больших команд разработчиков
- ▶ в критически важных системах реального времени с высокими требованиями по надежности
- ▶ при разработке новой версии уже существующего продукта или переносе его на новую платформу
- ▶ в организациях, имеющих опыт создания программных систем определенного типа (например, бухгалтерский учет, начисление зарплаты и пр.)



Итеративное наращивание возможностей системы



На каждой итерации строится программное изделие, которое в принципе может применяться на практике, пусть даже не до конца реализуя функциональность в целом.

Рост количества прямоугольников при переходе от итерации к итерации отражает поступление новых требований в ходе развития проекта

Примечание: пунктиром выделены отработанные требования



Причины использования эволюционных и инкрементных моделей жизненного цикла ПО

«Самой тяжелой составляющей процесса построения программной системы является принятие однозначного решения о том, что именно необходимо построить. Ни одна из остальных составляющих работы над концепцией не представляет собой такую трудность, как определение детальных технических требований, включая все аспекты соприкосновения продукта с людьми, машинами и другими программными системами. Ни одна другая составляющая работы в такой степени не наносит ущерб полученной в результате системе, если она выполнена неправильно. Именно эту составляющую процесса разработки тяжелее всего исправить на более поздних этапах. Следовательно, самая важная функция, которую выполняет разработчик клиентских программ, заключается в итеративном извлечении и уточнении требований к продукту. Ведь на самом деле клиент не имеет представления о том, что именно он хочет получить»

Ф. Брукс

Основные задачи, решаемые за счет создания прототипов:

- ▶ итеративное извлечение и уточнение требований к продукту
- ▶ возможность изменить эксплуатационное окружение программной системы
- ▶ снижение неопределенности (а следовательно, и рисков) по мере завершения каждой итерации



Эволюционные и инкрементные модели жизненного цикла

- ▶ Эволюционная модель быстрого прототипирования
- ▶ Модель быстрой разработки приложений RAD (Rapid Application Development)
- ▶ Инкрементная модель

Краткое описание подхода к разработке ПО: модели основываются на создании последовательности релизов и/или прототипов, критический анализ которых производится заказчиком. В процессе такого анализа уточняются требования к программному продукту

Данные модели чаще всего применяются в системах с ярко выраженным пользовательским интерфейсом



Достоинства эволюционных и инкрементных моделей ЖЦ

- ▶ Взаимодействие пользователя и/или заказчика с системой начинается на раннем этапе разработки, что минимизирует возможность их неудовлетворенности конечным продуктом и гарантирует соответствие программной системы коммерческим потребностям
- ▶ Благодаря раннему знакомству пользователя с системой сводится к минимуму количество неточностей в требованиях
- ▶ Модели позволяют выполнить несколько итераций на всех фазах жизненного цикла
- ▶ При использовании моделей заказчикам демонстрируются постоянные, видимые признаки прогресса в реализации проекта, что дает им возможность чувствовать себя более уверенно
- ▶ Благодаря меньшему объему доработок уменьшаются совокупные затраты на разработку и обеспечивается управление рисками
- ▶ Документация сконцентрирована на конечном продукте, а не на процессе его разработки



Недостатки итерационных моделей ЖЦ

- ▶ При использовании эволюционных прототипных моделей решение трудных проблем может отодвигаться на будущее. В результате конечный продукт может не оправдать тех надежд, которые возлагались на прототип
- ▶ На очередном итерационном этапе прототип представляет собой частичную реализацию системы, поэтому если выполнение проекта завершается досрочно, у конечного пользователя останется только лишь незавершенный продукт
- ▶ С учетом создания рабочих прототипов, качеству всего ПО или долгосрочной эксплуатационной надежности может бытьделено недостаточно внимания
- ▶ Разработанные прототипы часто страдают от неадекватной или недостающей документации
- ▶ Сложность завершения работы над проектом



Сочетание последовательного и эволюционного подхода: спиральная модель

Краткое описание	Достоинства	Недостатки	Примеры применения
Цикл разработки разбит на небольшие участки. Каждый участок представляет собой каскадный процесс, в котором выполняются обзор требований, обновление документации и некоторая разработка кода. Результат текущей итерации является входным значением для следующей. При этом программная система создается по частям с использованием метода прототипирования. После каждого витка спирали осуществляется анализ рисков	<ol style="list-style-type: none">Модель позволяет пользователям «увидеть» систему на ранних этапах разработки за счет использования ускоренного прототипированияМодель обеспечивает возможность разработки сложной программной системы «по частям», с выделением на первых этапах наиболее значимых требованийВ модели предусмотрена возможность гибкого проектирования, поскольку в ней воплощены преимущества каскадной модели, и в тоже время, разрешены итерации по всем фазам этой же моделиМодель предусматривает активное участие пользователей в работах по планированию, анализу рисков, разработке и оценке полученных результатовМодель позволяет контролировать риски	<ol style="list-style-type: none">Модель имеет усложненную структуру, что затрудняет ее применениеПри использовании модели часто возникает сложность анализа и оценки рисков при выборе вариантовСложность оценки точки перехода на следующий цикл спиралиСпираль может продолжаться до бесконечности, поскольку каждая ответная реакция заказчика на созданную версию может порождать новый цикл, что отдаляет окончание работы над проектом	<ol style="list-style-type: none">При разработке систем, требующих большого объема вычислений (например, системы, обеспечивающие принятие решений)При выполнении проектов в области аэрокосмической промышленности, обороны и инжиниринга, где уже имеется позитивный опыт ее использованияПри реализации крупных бизнес-проектов



Факторы, влияющие на выбор модели жизненного цикла

- ▶ ***характеристики требований:***

- ▶ являются ли требования легко определимыми и стабильными в процессе разработки
- ▶ требуется ли проверка концепции для демонстрации возможностей ПП

- ▶ ***характеристики участников команды разработчиков:***

- ▶ являются ли проблемы предметной области, а также инструменты, используемые в проекте новыми для большинства разработчиков
- ▶ изменяются ли роли участников проекта во время жизненного цикла

- ▶ ***характеристики коллектива пользователей:***

- ▶ будет ли присутствие пользователей ограничено в жизненном цикле
- ▶ будет ли заказчик отслеживать ход выполнения проекта

- ▶ ***характеристики типа проектов и рисков:***

- ▶ будет ли проект идентифицировать новое направление продукта для организации
- ▶ являются ли "прозрачными" интерфейсные модули
- ▶ должна ли обеспечиваться высокая степень надежности

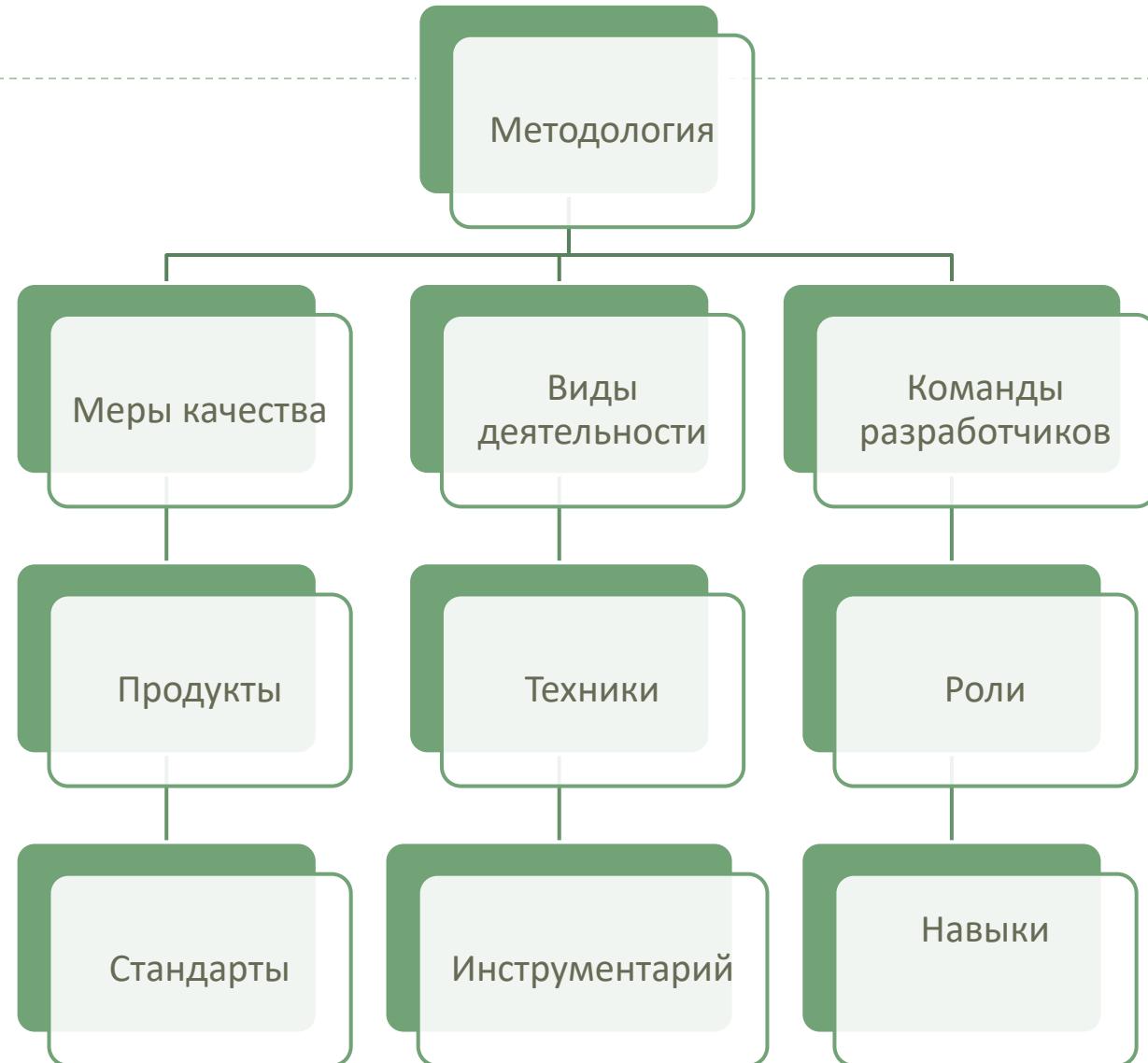


Методологии программирования

Методология – ряд
связанных между собой
методов или техник

Толковый словарь Webster

Примечание: все
составляющие методологии
должны рассматриваться в
контексте системы ценностей
команды разработчиков



Методологии программирования

Жесткие методологии	Гибкие методологии
Ориентация на предсказуемые процессы разработки программного обеспечения с четко обозначенными целями	Осознание того, что процессы разработки программного обеспечения в принципе непредсказуемы
Распознавание ситуаций и применение готовых методов	Распознавание ситуаций и конструирование методов для работы в них
Планирование, в котором определяются этапы с объемом работ, ресурсами, сроками и уровнем качества работ	Соблюдение баланса между параметрами проекта: объем работ, ресурсы, сроки и уровень качества работ
Заказчик — внешний по отношению к проекту субъект, влияющий на разработку только через предоставление ресурсов и контроль результатов, в том числе по поэтапным срокам выполнения проекта	Заказчик (его представитель) — член команды разработчиков, наделенный правом влиять на разработку; его главной целью является отслеживание актуальности решаемых задач
Ролевое разделение труда работников проекта	Совместная деятельность сотрудников и деперсонифицированная ответственность
Дисциплина и подчинение	Самодисциплина и сотрудничество
Обезличенный процесс, исполнители которого определяются только по квалификационным требованиям	Процесс, максимально учитывающий личностные качества исполнителей



Качество программного обеспечения

- ▶ Определение ISO: Качество - это полнота свойств и характеристик продукта, процесса или услуги, которые обеспечивают способность удовлетворять заявленным или подразумеваемым потребностям
- ▶ Определение IEEE: Качество программного обеспечения - это степень, в которой оно обладает требуемой комбинацией свойств

Стандарты качества ПО:

- ▶ ISO/IEC 9126:1-4:2002 (ГОСТ Р ИСО/МЭК 9126-93)
- ▶ ISO/IEC 14598 – набор стандартов, регламентирующий способы оценки характеристик качества

В совокупности они образуют модель качества - SQuaRE (Software Quality Requirements and Evaluation)



Качество программного обеспечения

- ▶ внешнее качество, заданное требованиями заказчика в спецификациях и отражающееся характеристиками конечного продукта;
- ▶ внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапах жизненного цикла ПС;
- ▶ качество при использовании в процессе нормальной эксплуатации и результативность достижения потребностей пользователей с учетом затрат ресурсов (эксплуатационное качество)

Качество объекта зависит от того, для какой **цели**, для какого **потребителя** и для каких **условий** делается его оценка



Стандарт ISO/IEC 9126:1-4: Информационная технология - Качество программных средств

- ▶ Часть 1: Модель качества
- ▶ Часть 2: Внешние метрики качества
- ▶ Часть 3: Внутренние метрики качества
- ▶ Часть 4: Метрики качества в использовании

6 базовых характеристик качества, используемых в модели стандарта ISO/IEC 9126-1

функциональная пригодность	практичность
надежность	сопровождаемость
эффективность	мобильность



Набор стандартов ISO 9000

- ▶ ISO 9000:2000 Quality management systems – Fundamentals and vocabulary.
Системы управления качеством – Основы и словарь.
- ▶ ISO 9001:2000 Quality management systems – Requirements. Models for quality assurance in design, development, production, installation and servicing.
Системы управления качеством – Требования. Модели для обеспечения качества при проектировании, разработке, производстве, установке и обслуживании. Определяет общие правила обеспечения качества результатов во всех процессах жизненного цикла

Набор стандартов ISO 9000 регулирует общие принципы обеспечения качества процессов производства во всех отраслях экономики



Модель СММ (5 уровней зрелости организации, занимающейся разработкой ПО)

5. Оптимизированный

- Постоянное улучшение процессов
- Управление изменениями
- Предотвращение дефектов

4. Управляемый

- Управление процессом разработки ПО на основе количественных критериев качества
- Ориентация на инновации (методы, технологии, инструментарий)

3. Определенный

- Экспертная оценка программ
- Межгрупповая координация
- Повышение квалификации сотрудников
- Формализованное определение процесса

2. Повторяемый

- Управление требованиями
- Управление конфигурацией
- Планирование и отслеживание проекта
- Обеспечение качества ПО

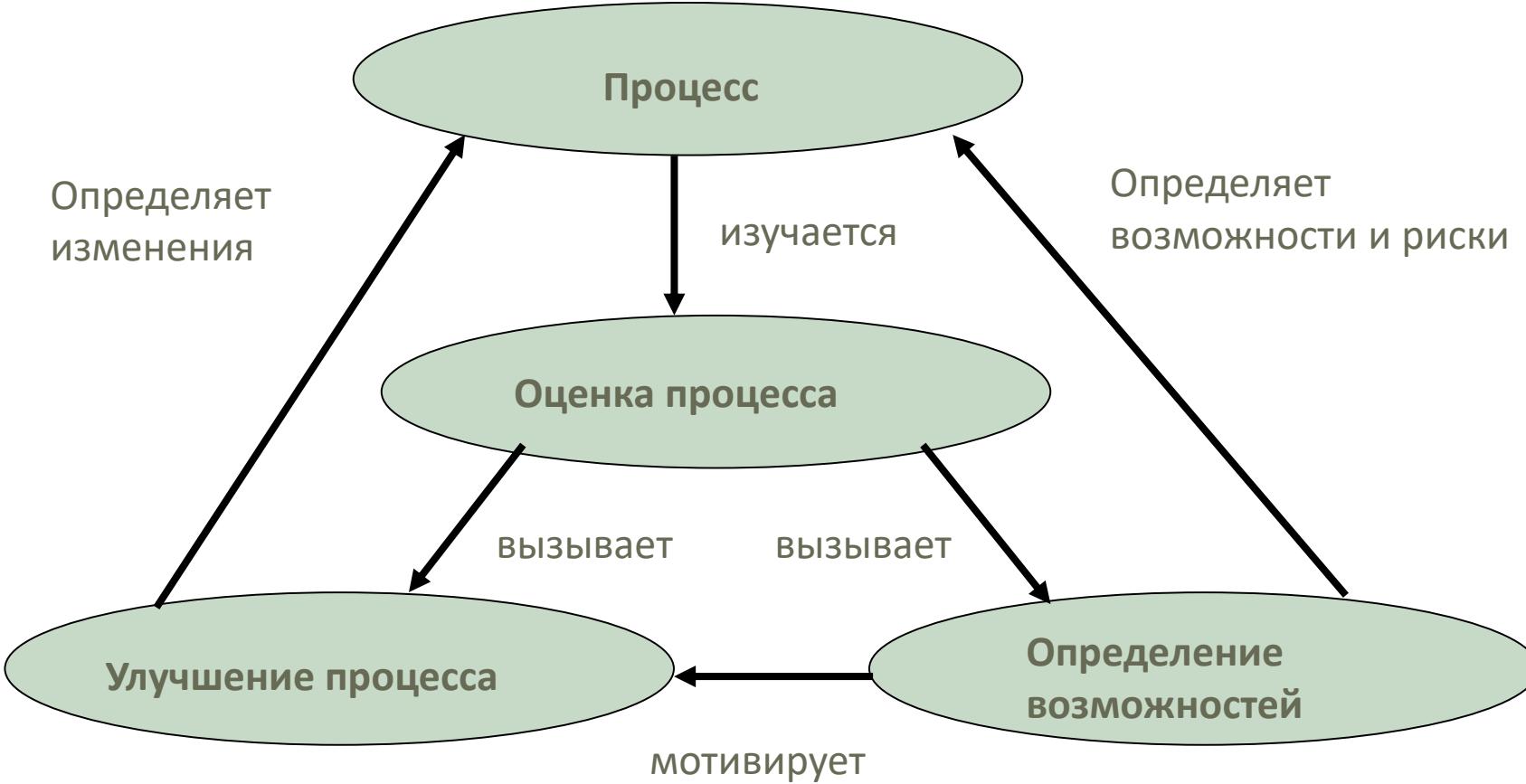
1. Начальный

- Непредсказуемое качество процесса
- Индивидуальные решения для каждого проекта

Ограничения на применение модели СММ:

- ▶ стандарт СММ является собственностью Software Engineering Institute (SEI) и не является общедоступным (в частности, дальнейшая разработка стандарта ведется самим институтом, без заметного влияния программистского сообщества)
- ▶ оценка качества процессов организаций может проводиться только специалистами, прошедшиими специальное обучение и SEI;
- ▶ стандарт ориентирован на применение в относительно крупных компаниях

Основные элементы стандарта ISO/IEC 15504 (SPICE)



- **Оценка процесса** происходит путем сравнения процесса разработки ПО, существующего в данной организации, с описанной в стандарте моделью.
- **Определение возможностей** процесса происходит на основе выбора одного из альтернативных путей перехода к более совершенному процессу
- **Улучшение** того или иного **процесса** происходит на основе сформулированных на предыдущих этапах целей и направлено на реализацию выбранной стратегии перехода



Модель Гантера: модель фазы-функции

Модель жизненного цикла программного обеспечения может использоваться для комплексного управления программным проектом, если наложить на нее контрольные точки, задающие организационно-временные рамки проекта, и организационно-технические, так называемые производственные функции, которые выполняются при его развитии

Этот подход реализован в модели Гантера в виде матрицы «фазы — функции»

Модель Гантера имеет два измерения:

- ▶ фазовое, отражающее этапы выполнения проекта и сопутствующие им события
- ▶ функциональное, показывающее, какие производственные функции выполняются в ходе развития проекта, и какова их интенсивность на каждом из этапов



Модель фазы-функции (фазовое измерение)



Основные этапы и контрольные точки

- ▶ *Этап исследования* — для проекта обосновываются необходимые ресурсы (контрольная точка 1) и формулируются требования к разрабатываемому изделию (контрольная точка 2)
- ▶ *Анализ осуществимости* — определяется возможность конструирования программного средства с технической точки зрения (достаточно ли ресурсов, квалификации персонала и т.п.), решаются вопросы экономической и коммерческой эффективности
- ▶ *Конструирование* — начинается обычно на этапе анализа осуществимости, как только документально зафиксированы предварительные цели проекта (контрольная точка 2), и заканчивается утверждением проектных решений в виде официальной спецификации на разработку (контрольная точка 5)
- ▶ *Программирование* — программирование компонентов с последующей сборкой системы. Этап завершается, когда разработчики заканчивают документирование, отладку и компоновку и передают ПС службе, выполняющей независимую оценку результатов работы (независимые испытания начались — контрольная точка 7)
- ▶ *Оценка* — является буферной зоной между началом испытаний и практическим использованием изделия. Этап начинается, как только проведены внутренние (силами разработчиков) испытания изделия (контрольная точка 6) и заканчивается, когда подтверждается готовность изделия к эксплуатации (контрольная точка 9)
- ▶ *Использование* — начинается в момент передачи изделия на распространение (контрольная точка 8). Этап продолжается, пока изделие интенсивно эксплуатируется. Он связан с внедрением, обучением, настройкой и сопровождением, возможно, с модернизацией изделия. Этап заканчивается, когда разработчики прекращают систематическую деятельность по сопровождению и поддержке данного программного изделия (контрольная точка 10)



Модель фазы – функции (функциональное измерение)



Основные функции

- ▶ *Планирование* — содержание того, что должно планироваться на каждом из этапов определяется по характеру контрольных точек. Конкретные методы планирования определяются концепциями, принимаемыми для данного проекта
- ▶ *Разработка* — эта функция пронизывает весь проект и каждый раз направлена на получение рабочего продукта этапа
- ▶ *Обслуживание* — функция, обеспечивающая максимально комфортную обстановку для выполнения некоторой деятельности. По интенсивности обслуживания можно косвенно судить о качестве организации работ проекта
- ▶ *Выпуск документации*. Документация в проекте рассматривается как полноправный вид рабочих продуктов, сопровождающий другие рабочие продукты: программы, диаграммы моделей системы и прочее
- ▶ *Испытания*. Испытывать приходится все рабочие продукты проекта
- ▶ *Поддержка использования рабочих продуктов* — это функция, выполнение которой необходимо в связи с передачей продукта в эксплуатацию. Содержание ее связано с обучением и созданием необходимой для использования продуктов инфраструктуры
- ▶ *Сопровождение* (по Гантеру) отличается от поддержки тем, что оно организуется для внешнего использования продуктов





Спасибо за внимание!



Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана

baryshnikovam@mail.ru

Лекция 4

Инициирование программного проекта. Концепция проекта

Если бы мне разрешили разработать только один документ, модель или другой артефакт для поддержки программного проекта, я бы выбрал краткий, хорошо сформулированный документ-концепцию

Филипп Крачтен



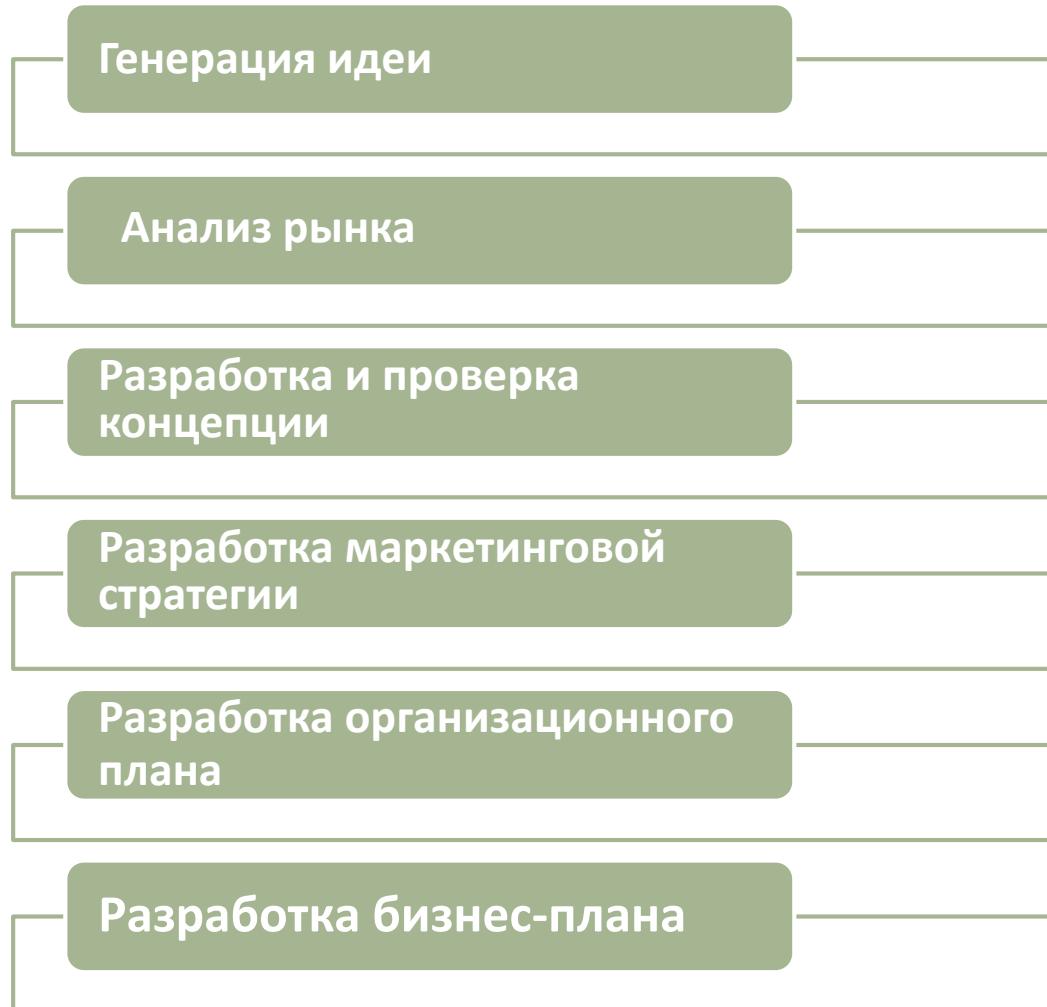
Фаза инициации: если совсем коротко

Инициация проекта – это важный этап, от которого напрямую зависит, будут ли вообще предприниматься усилия для его реализации. Вам нужно убедить людей, принимающих решение (прежде всего - инвесторов), в целесообразности запуска проекта. Они должны поверить в придуманную вами историю и ее **ценность**. В этом помогает любая информация, которая доносит эту ценность прежде всего как ответ на два основных вопроса: «зачем» и «как». Именно эта информация и составляет основу концепции проекта

- ▶ **Описание ценности.** Опишите проблему и возможности, достигаемые бизнесом цели и другие предположения о том, что будет происходить. Бывает полезно показать на контрасте варианты того, что может быть достигнуто в результате реализации проекта (возможность), и что будет если его вообще не начинать (риски, проблемы)
- ▶ **Демонстрация альтернатив.** Любую проблему можно решить разными способами. Покажите, почему ваше решение является оптимальным, почему оно принесет больше ценности и/или несет в себе меньше рисков
- ▶ **Требования к проекту.** Опишите, как вы видите реализацию проекта, что в нем должно быть, а чего быть не должно
- ▶ **План реализации.** Ответьте на вопрос, как проект должен реализоваться, укажите основные этапы реализации, определите период реализации и выполните укрупненный анализ финансовых потребностей



Фаза инициации в жизненном цикле программного проекта



Основной целью фазы инициации является подтверждение актуальности и осуществимости идеи, ее коммерческой привлекательности и принятие решения о начале процесса разработки продукта

Ее результатом является разработка бизнес-плана программного проекта

Бизнес-план — это форма представления деловых предложений и проектов, содержащая развернутую информацию о производственной, сбытовой и финансовой деятельности организации и оценку перспектив, условий и форм сотрудничества на основе баланса собственного экономического интереса фирмы и интересов партнеров, инвесторов, потребителей, посредников и других участников инвестиционного проекта



Источники идей программных продуктов



Идея программного продукта, который компания могла бы предложить рынку, формируется в виде описания ценностного предложения для потребителя, состоящего из нескольких предложений, без описания конкретной формы реализации

Большую роль в принятии решения о перспективности предложенной идеи играют первичные рыночные исследования. На данной стадии важно понять, действительно ли существует проблема, которую команда разработчиков собирается решать при помощи своего продукта, поэтому обсуждение и оценка идеи должны быть проведены с представителями потенциальной целевой аудитории в виде опроса, собеседования или фокус-группы



Структура бизнес-модели программного продукта



Бизнес-модель по Остервальдеру и Пинье

Разработка концепции программного проекта должна начинаться с выявления всего множества потенциальных потребителей, которым может быть интересен продукт в различных его вариантах:

- ▶ в аспекте его функциональности
- ▶ с точки зрения способов оказания услуг, отвечающих запросам потребителей
- ▶ в аспекте вопросов лицензирования и поддержки продукта



Концепция проекта

Концепция (от лат. *conceptio* — понимание, система), определенный способ понимания, трактовки какого-либо предмета, явления, процесса, основная точка зрения на предмет и др., руководящая идея для их систематического освещения

- ▶ Концепция проекта разрабатывается на основе анализа потребностей бизнеса
- ▶ Она необходима для согласования единого видения целей, задач и результатов всеми участниками проекта
- ▶ Концепция определяет ***что и зачем*** делается в проекте
- ▶ Концепция не затрагивает развитие проекта в глубину, а дает максимально широкий охват проблем и определяет долгосрочные перспективы продукта
- ▶ Концепция проекта - это ключевой документ, который используется для принятия решений в ходе его реализации, а также обеспечивает возможность подтверждения результата на фазе приемки

Любая концепция по своей сути – это соглашение между членами определенной группы людей, т.е. ее можно рассматривать как своего рода коллективное одобрение. **Достичь соглашения между людьми с разнонаправленными целевыми установками чрезвычайно трудно, работать без него – неэффективно!!!**



На какие вопросы должна ответить концепция проекта

- ▶ Каковы предпосылки реализации именно этого проекта?
- ▶ Что подтверждает необходимость его реализации?
- ▶ Какова степень технической реализуемости идеи?
- ▶ Каково внешнее окружение идеи?
- ▶ В чем заключаются основные цели?
- ▶ Каковы укрупненные финансовые условия реализации проекта?
- ▶ Какие ресурсы (помимо денег) необходимы для его реализации?
- ▶ Есть ли четкое понимание времени старта?

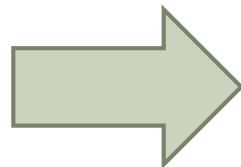
Какие действия выполняются при поиске ответов на эти вопросы

- ▶ Определение способа реализации идеи
- ▶ Фиксация ожиданий от заинтересованных сторон
- ▶ Укрупненный анализ ресурсов, необходимых для достижения установленных целей
- ▶ Анализ рисков и допущений
- ▶ Назначение ответственных за реализацию
- ▶ Определение численности и состава команды проекта
- ▶ Формирование укрупненного списка процессов
- ▶ Формирование общего плана управления проектом



Структура концепции

Концепция должна содержать изложение вопросов, которые помогут глубже понять содержание проектной идеи и оценить ее практичесность и эффективность реализации, поэтому в описание концепции обычно (но не обязательно) входят следующие разделы

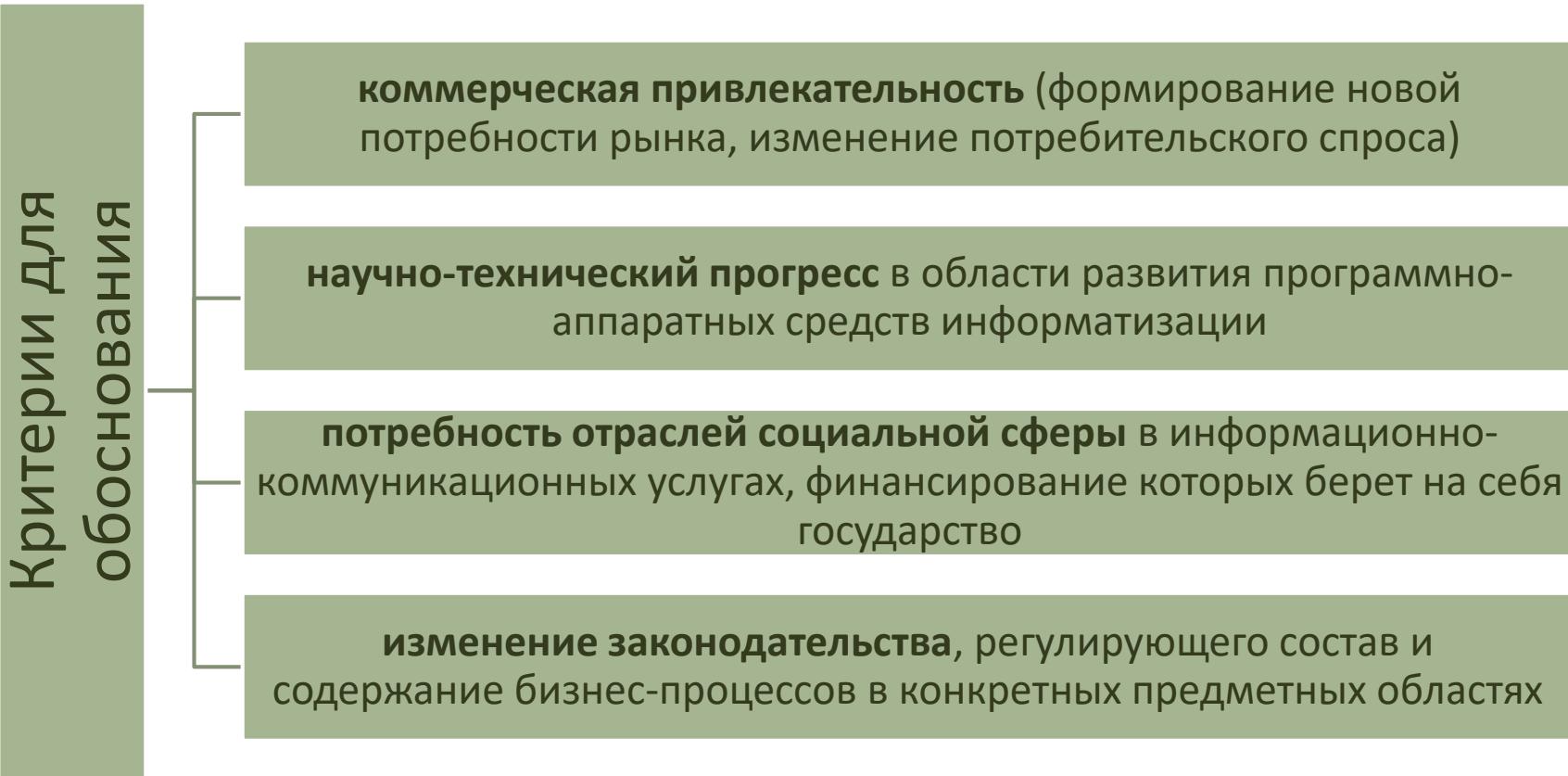


- ▶ Название проекта
- ▶ Бизнес-обоснование потребности или необходимости в разработке ПП
- ▶ Цели проекта
- ▶ Результаты проекта
- ▶ Допущения и ограничения
- ▶ Содержание проекта
- ▶ Ключевые участники и заинтересованные стороны
- ▶ Экономика проекта
- ▶ Ресурсы проекта
- ▶ Сроки
- ▶ Риски
- ▶ Критерии приемки

Примечание: документ-концепцию иногда называют Уставом проекта



Бизнес-обоснование потребности или необходимости в разработке программного продукта



На что делать упор при обосновании?

Коммерческая привлекательность	Уникальность	Стратегия продвижения
какие проблемы существуют у потенциального заказчика, насколько значимо для него решение данных проблем	обладает ли продукт какими-либо новыми уникальными особенностями	ваша целевая аудитория: кому вы собираетесь предлагать ваш продукт или услугу
зачем нужен данный продукт, какова его основная идея, какие требования к ПП могут предъявлять потенциальные пользователи	чем отличается ваш продукт от продуктов конкурентов	каким образом вы собираетесь продавать продукт: возможные каналы поставки продукта, способ организации взаимоотношений с пользователями
какой полезный эффект может извлечь потенциальный потребитель от использования продукта и сколько времени уйдет на его разработку	с кем вы собираетесь конкурировать в выбранных сегментах рынка, кто является производителем аналогичных продуктов, за какую цену продаются эти аналоги	как вы собираетесь привлечь покупателей



Цели проекта

Цели проекта должны отвечать на вопрос, зачем данный проект нужен. Они должны описывать бизнес-потребности и задачи, которые решаются в результате реализации проекта

Примеры целей



Изменения в компании

Например, автоматизация ряда бизнес-процессов для повышения эффективности основной производственной деятельности



Реализация стратегических планов

Например, завоевание значительной доли растущего рынка за счет вывода на него нового продукта



Выполнение контрактов

Например, разработка программного обеспечения по заказу



Разрешение специфических проблем

Например, доработка программного продукта в целях приведения его в соответствие с изменениями в законодательстве

Требования к целям:

- ▶ значимость (направленность на достижение стратегических целей компании)
- ▶ конкретность (учет специфических особенностей конкретного проекта)
- ▶ измеримость (наличие проверяемых количественных оценок)
- ▶ реалистичность (достижимость)



Результаты проекта

Результаты проекта отвечают на вопрос, **что** должно быть получено после его завершения. Результаты проекта должны определять:

- ▶ какие именно бизнес-выгоды получит заказчик в результате проекта
- ▶ что конкретно (какой продукт или услуга) станет итогом его реализации

По отношению к результатам должны быть приведены высокоуровневые требования, либо краткое описание и при необходимости – ключевые свойства и/или характеристики продукта / услуги

Очень важно чтобы результаты проекта были измеримыми, т.е. при их оценке должна иметься возможность сделать заключение, достигнуты оговоренные в концепции требования или нет



Жизненный цикл проекта и продукта

Жизненный цикл
проекта



Жизненный цикл
продукта

Продукт может быть создан:

- ▶ в результате реализации проекта (если является одной из его целей)
- ▶ в процессе его реализации (если продукт является одним из этапов достижения каких-либо из его целей)
- ▶ до старта (если его существование является условием для осуществления проекта)
- ▶ после завершения (если он не является целью)



Содержание проекта

- ▶ Описывается в виде функциональных и нефункциональных требований к программному продукту, определяемых на данный момент времени
- ▶ Функциональные требования должны отражать потребности потенциальных пользователей, а нефункциональные — характеристики качества ПП
- ▶ Уровень детализации требований должен быть достаточным для представления и краткого описания архитектуры будущего программного продукта как совокупности программных модулей (компонентов) с перечислением их функционала



Ключевые участники и заинтересованные стороны

К ключевым участникам программного проекта, как правило, относятся:

- ▶ *инвестор или спонсор проекта* - лицо или группа лиц, предоставляющая финансовые ресурсы на различных условиях
- ▶ *заказчик проекта* - лицо или организация, которые будут использовать продукт, услугу или результат проекта. Следует учитывать, что заказчик и инвестор проекта не всегда совпадают
- ▶ *куратор проекта* - представитель исполнителя, уполномоченный принимать решение о выделении ресурсов и изменениях в проекте
- ▶ *руководитель проекта* - представитель исполнителя, ответственный за реализацию проекта в срок, в пределах бюджета и с заданным качеством
- ▶ *соисполнители проекта* – как правило, это субподрядчики и поставщики

Для определения основных интересантов важно определить тип рынка, на который выводится ПО (промышленный или потребительский) и привести его описание с выделением множества групп потенциальных потребителей, которым может быть интересен продукт, отвечающий их запросам



Экономика программного проекта

В данном разделе концепции приблизительно оцениваются:

- ▶ трудозатраты на разработку программных продуктов
- ▶ бюджет проекта
- ▶ рыночная цена продажи одной лицензии
- ▶ минимально допустимый объем продаж, покрывающий расходы бюджета

Минимально допустимое количество продаж, покрывающее все затраты на разработку первой версии программного продукта, его продвижение на рынок и поставку потребителям, не принося при этом ни прибыли, ни убытков, называется «точка безубыточности»



Экономика программного проекта

Основным методом определения точки безубыточности является **CVP-анализ** (Cast Value Profit — затраты, объем, прибыль), основанный на оценке соотношений затрат, выручки и прибыли:

$$t_b = \frac{a}{(s * x - b * x) * s * x}$$

где:

x — количество продаж

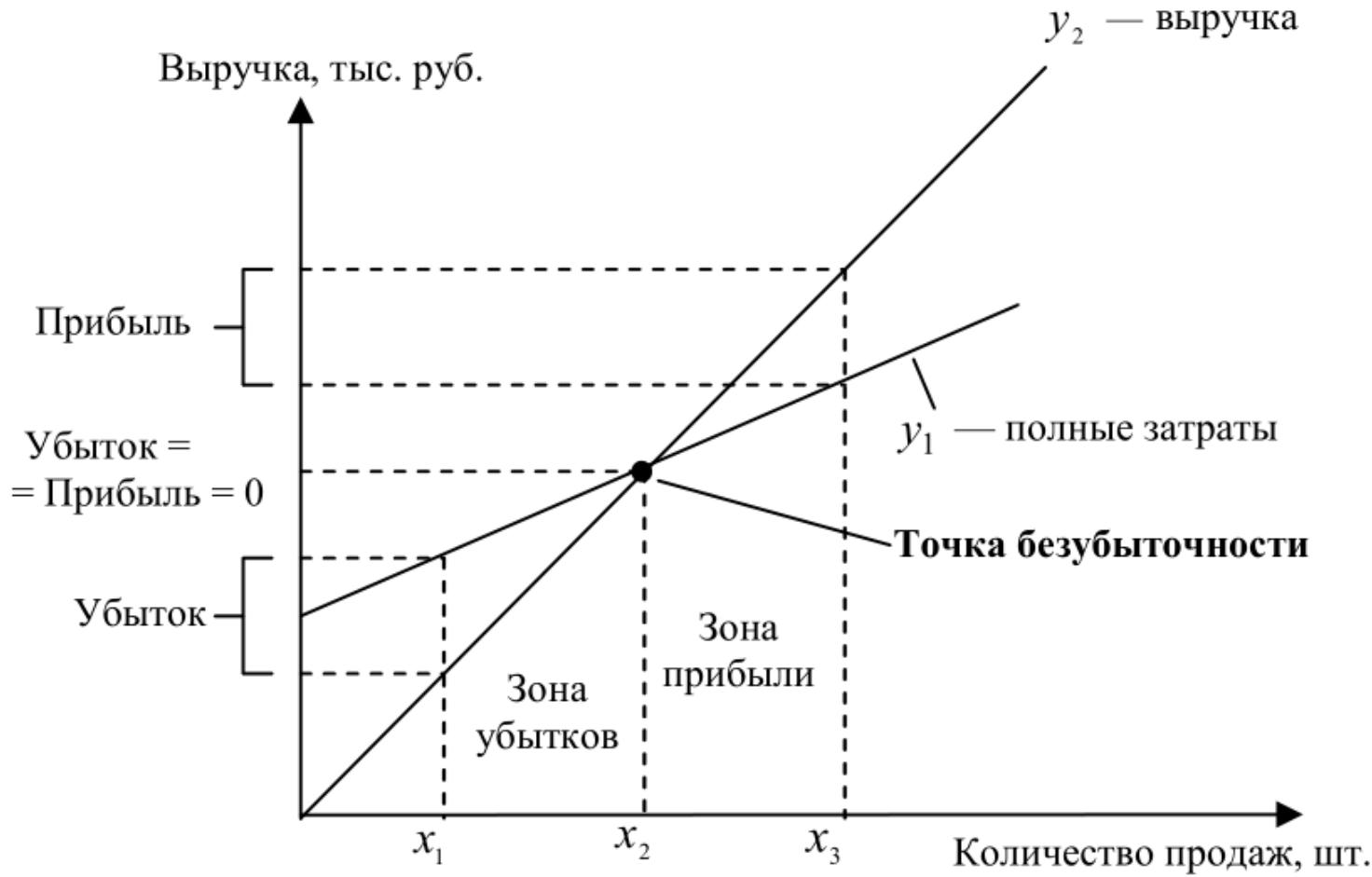
s — рыночная цена продажи единицы продукции

a — величина фиксированных расходов

b — величина переменных издержек на единицу продукции



Графическая интерпретация определения и анализа точки безубыточности



Экономика программного проекта

Количество продаж, при котором достигается точка безубыточности (прибыль фирмы равна нулю): $x_0 = \frac{a}{s-b}$

Если известен объем рынка, то рыночная цена продажи одной лицензии на ПП при нулевом уровне прибыли: $s_0 = \frac{a+b*x_0}{x_0}$

Количество продаж при заданном уровне прибыли P_0 и рыночной цене s_0 :

$$X_p = \frac{P_0 + a}{s_0 - b}$$

Чистая прибыль: $P = s*x - (a + b*x) = (s - b) * x - a$



Ресурсы

Ресурсы необходимые для реализации программного проекта:

- ▶ людские ресурсы и требования к квалификации персонала
- ▶ оборудование, услуги, расходные материалы, лицензии на ПО
- ▶ бюджет проекта, план расходов и, при необходимости, предполагаемых доходов проекта с разбивкой по статьям и фазам/этапам проекта

Необходимо помнить, что трудозатраты, связанные непосредственно с программированием, в программном проекте составляют примерно **25%**. Поэтому если по предварительной оценке для реализации требуемой функциональности необходимо написать 10 KSLC (тысяч строк исходного программного кода), а программисты пишут в среднем по 100 SLOC в день, то общие трудозатраты на проект будут **не 100 чел.*дней**, а не менее чем **400 чел.*дней**. Остальные ресурсы потребуются на анализ и уточнение требований, проектирование, документирование, тестирование и другие проектные работы



Формула Барри Боэма для оценки длительности программного проекта на основе общей трудоемкости

Если трудоемкость проекта составляет N ч.м. (человеко-месяцев), то можно утверждать что:

- ▶ Существует оптимальное, с точки зрения затрат, время выполнения графика для первой поставки: $T_{\text{опт}} = 2,5 (N \text{ ч.м.})^{1/3}$. То есть оптимальное время в месяцах пропорционально кубическому корню предполагаемого объема работ в человеко-месяцах
- ▶ Кривая стоимости медленно растет, если запланированный график длиннее оптимального. Работа занимает все отведенное для нее время
- ▶ Кривая стоимости резко растет, если запланированный график короче оптимального. Практически ни один проект невозможно завершить быстрее, чем за $\frac{3}{4}$ расчетного оптимального графика вне зависимости от количества занятых в нем людей!

Для программного проекта недостаточно определить только срок его завершения. Необходимо еще определить его этапы и контрольные точки, в которых будет происходить переоценка проекта на основе реально достигнутых показателей, причем каждая контрольная точка должна характеризоваться датой и объективными критериями ее достижения



Риски

Риск - неопределенное событие или условие, наступление которого отрицательно или положительно сказывается на целях проекта

На этапе инициации, когда нет необходимых данных для проведения детального анализа, часто приходится ограничиваться качественной оценкой общего уровня рисков: низкий, средний, высокий

Критерии приемки

Критерии приемки должны определять числовые значения характеристик системы, которые должны быть продемонстрированы по результатам приемо-сдаточных испытаний или опытной эксплуатации и однозначно свидетельствовать о достижении целей проекта



Управление приоритетами проектов

В компании, которая принимает решение о старте того или иного проекта разработки ПО, должна существовать единая система критериев для оценки его значимости

Поэтому оценка концепции любого проекта целесообразно проводить с использованием трех характеристик:

- ▶ финансовой ценности
- ▶ стратегической ценности
- ▶ уровня рисков

Если компания уделяет мало внимания управлению приоритетами своих проектов, то это приводит к переизбытку реализуемых проектов, перегруженности исполнителей, постоянным авралам и сверхурочным работам и, как следствие, к низкой эффективности производственной деятельности



Финансовая ценность проекта

- ▶ *Высокая.* Ожидаемая окупаемость до 1 года. Ожидаемые доходы от проекта не менее чем в 1.5 раза превышают расходы. Все допущения при проведении этих оценок четко обоснованы
- ▶ *Выше среднего.* Ожидаемая окупаемость проекта от 1 года до 3 лет. Ожидаемые доходы от проекта не менее чем в 1.3 раза превышают расходы. Большинство допущений при проведении этих оценок имеют под собой определенные основания
- ▶ *Средняя.* Проект позволяет улучшить эффективность производства и потенциально может снизить расходы компании не менее чем на 30%. Проект может иметь информационную ценность или помочь лучше контролировать бизнес
- ▶ *Низкая.* Проект снижает расходы компании не менее чем на 10% и дает некоторые улучшения производительности производства

Одной финансовой ценности для определения приоритета проекта недостаточно, он также должен соответствовать стратегическим целям компании



Стратегическая ценность проекта

- ▶ *Высокая.* Проект обеспечивает стратегическое преимущество, дает устойчивое увеличение рынка или позволяет выйти на новый рынок. Он решает проблемы, общие для большинства клиентов компании. Повторение проекта конкурентами затруднено или потребует от 1 до 2 лет
- ▶ *Выше среднего.* Проект создает временные конкурентные преимущества. За счет его реализации обеспечивается возможность выполнения обязательств перед многими клиентами компании. Конкурентное преимущество может бытьдержано в течение 1 года
- ▶ *Средняя.* Поддерживается доверие рынка к компании. Проект повышает мнение клиентов о качестве предоставляемых услуг или способствует выполнению обязательств перед несколькими клиентами. Конкуренты уже имеют или способны повторить новые возможности в пределах года
- ▶ *Низкая.* Стратегическое воздействие отсутствует или незначительно. Влияние на клиентов несущественно. Конкуренты могут легко повторить результаты проекта



Уровень рисков проекта

- ▶ *Низкий.* Цели проекта и требования хорошо поняты и документированы. Масштаб и рамки проекта заданы четко. Ресурсы требуемой квалификации доступны в полном объеме. Разрабатываемая система не потребует новой технологической платформы
- ▶ *Средний.* Цели проекта определены более-менее четко. Имеется хорошее понимание требований к системе. Масштаб и рамки проекта заданы достаточно четко. Ресурсы требуемой квалификации в основном доступны. Система создается на новой, но стабильной технологической платформе.
- ▶ *Выше среднего.* Цели проекта недостаточно четки. Задачи системы или бизнес-приложения поняты недостаточно полно. Понимание масштаба и рамок проекта недостаточно. Ресурсы требуемой квалификации сильно ограничены. Система создается на новой технологической платформе, которую можно охарактеризовать как нестабильную с точки зрения рынка
- ▶ *Высокий.* Цели проекта нечетки. Основные функциональные компоненты системы не определены. Масштаб и рамки проекта непонятны. Ресурсы требуемой квалификации практически отсутствуют. Система создается на новой технологической платформе, в отношении которой крайне мало ясности. Технологии имеют неподтвержденную стабильность



Резюме

- ▶ **Инициация проекта** – это момент, когда еще можно остановиться и подумать, а делать ли проект вообще? Когда еще не затрачены деньги, время и усилия. Когда вашей репутации хорошего руководителя проектов еще не нанесен урон, который потом будет сложно исправить. После того, как концепция проекта утверждена, а его руководитель назначен, что-то решать уже поздно
- ▶ Стадия **инициации** проекта – это возможность сделать так, чтобы основные заинтересованные лица поняли, о чем этот проект и какую роль они в нем играют

Причин инициации проекта может быть всего три: увеличение прибыли, сокращение расходов, снижение рисков



Кейс №1: Создание Интернет-магазина по продаже книг

Причины инициации проекта	Проведенные маркетинговые исследования показали, что наличие собственного интернет-магазина издательства увеличит прибыль на x%
Цель проекта	Создать интернет-магазин для продажи книг издательства Обеспечить популярность интернет-магазина Обеспечить постоянный поток заказов, проходящих через интернет-магазин
Задачи проекта	Подготовить техническое задание на разработку интернет-магазина Разработать интернет-магазин для продажи книг издательства Организовать доставку заказов интернет-магазина Провести рекламную кампанию для интернет-магазина Интегрировать интернет-магазин с платежными системами

Для формулирования целей и задач были выполнены следующие действия:

- Выявление потребностей в дополнительных рынках сбыта
- Оценка интернет-рынка книг по объему и стоимости реализации
- Определение себестоимости содержания интернет-магазина



Результаты проекта

Результат

установленная и настроенная система интернет-магазина;
количество посетителей не менее **y** в день;
количество заказов не менее **z** в день

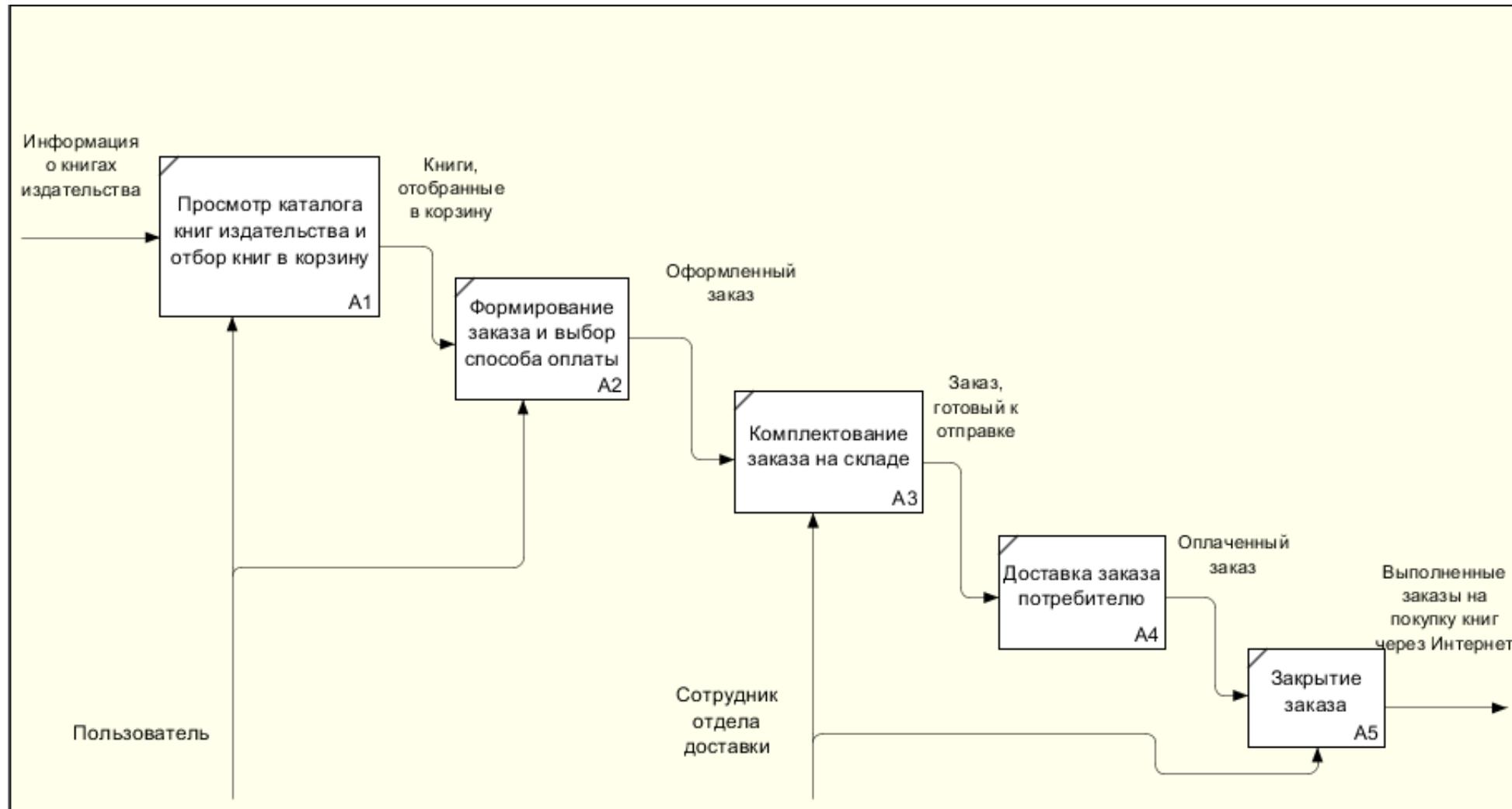


Краткое описание системы

- ▶ Система интернет-магазин предназначена для отображения списка публикуемых издательством книг с возможностью заказа и доставки необходимых изданий
- ▶ Пользователь интернет, зайдя на сайт интернет-магазина, должен иметь возможность ознакомиться со структурированным по категориям списком книг издательства. Для каждой книги должна быть возможность просмотреть подробную информацию. Пользователь может положить любую книгу из списка в корзину
- ▶ Когда корзина будет сформирована, пользователь может оформить заказ. При оформлении заказа пользователь должен иметь возможность выбрать удовлетворяющий его способ оплаты — наличными при доставке или банковской карточкой при оформлении заказа
- ▶ Оформленный заказ должен поступить в отдел доставки, где должен пройти этап сбора заказанных книг из имеющихся на складе (формирование заказа на складе), этап согласования доставки и этап доставки заказа конечному потребителю. После доставки заказ должен быть закрыт



Функциональная модель



Бизнес-требования

Требование	Описание
Просмотр списка книг	Интернет-пользователь должен иметь возможность ознакомиться со структурированным по категориям списком книг издательства, включая краткое описание каждой книги
Просмотр описания книги	Интернет-пользователь должен иметь возможность просмотреть полное описание книги
Добавление книги в корзину	Интернет-пользователь должен иметь возможность добавить в корзину нужную книгу из списка книг или из открытого описания книги
Оформление заказа	Интернет-пользователь должен иметь возможность оформить заказ на основе содержимого корзины. При оформлении заказа пользователь должен иметь возможность выбрать удовлетворяющий его способ оплаты — наличными при доставке или банковской карточкой при оформлении заказа



Бизнес-требования

Требование	Описание
Формирование заказа на складе	Сотрудник отдела доставки должен иметь возможность просматривать новые заказы и отправлять их на внутреннюю обработку и формирование комплекта книг для доставки из имеющихся на складе
Доставка заказа	Сотрудник отдела доставки должен иметь возможность распечатать бланк доставки заказа, связаться с клиентом, осуществить доставку и принять оплату наличными
Закрытие заказа	Сотрудник отдела доставки после осуществления доставки должен отметить факт оплаты наличными и закрыть заказ



Системные требования

Требование	Описание
Требование к оборудованию	Система должна располагаться на одном типовом сервере
Требование к производительности	Интернет-магазин должен обрабатывать не менее a пользователей в день и не менее b заказов в день



Требования к документации

Требование	Описание
Должна быть документация внутреннего пользователя	Для пользователей внутри компании должны присутствовать документы, описывающие бизнес-процесс работы с системой
Должны быть интерактивные подсказки на сайте	На сайте интернет-магазина должны присутствовать интерактивные подсказки, помогающие пользователю правильно заполнить поля





Спасибо за внимание!



Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана
Каф. ИУ-7

baryshnikovam@mail.ru

Лекция 5

Планирование программного проекта.

Структурная декомпозиция работ (WBS) как основа планирования проекта. Критический путь проекта

Провал планирования – это планирование провала.
Если Вы не знаете, куда направляетесь, то как Вы
узнаете, когда придете туда?



Сущность планирования

достижение целей проекта за счет формирования комплекса работ (мероприятий или действий), которые должны быть выполнены, определения способов выполнения данных работ, увязки необходимых для этого ресурсов и согласования действий участников проекта

Цель планирования

выбор способа создания программного продукта, который позволит выполнить техническое задание, соблюсти условия контракта, а также обеспечить уровень качества, соответствующий заданным требованиям

Планирование должно обеспечивать реализуемость проекта в заданные сроки с минимальной стоимостью, в рамках нормативных затрат ресурсов и с надлежащим качеством



На какие вопросы позволяет ответить планирование

- ▶ Что необходимо делать?
- ▶ Кто и что должен делать?
- ▶ Кто с кем взаимодействует?
- ▶ Когда и что должно быть сделано?
- ▶ Сколько и каких ресурсов нужно и для чего?
- ▶ Когда и откуда ресурсы должны поступать?
- ▶ Что сколько стоит?
- ▶ Что и когда должно быть оплачено?
- ▶ Какие это средства, каков их источник?
- ▶ Каковы лимиты ресурсов и бюджета?
- ▶ Какое требуется качество?
- ▶ Каковы риски проекта?
- ▶ Что выполнено на рассматриваемый момент, что нет?
- ▶ Кем и какие нарушены сроки?
- ▶ Что необходимо предпринять, чтобы проект был выполнен вовремя?

Планы ничего не значат. Но сам процесс планирования всеобъемлющ

Д. Эйзенхауэр



Принципы планирования в проекте

- ▶ **Целенаправленность.** Планирование рассматривается как процесс развертывания главной цели проекта в иерархическую последовательность задач до уровня отдельных мероприятий, действий и работ с определением порядка их выполнения
 - ▶ **Комплексность** планирования означает полный охват научных, проектных, организационных, производственных и других мероприятий и работ, направленных на достижение результатов проекта
 - ▶ **Сбалансированность по ресурсам** означает, что планы не содержат задач и работ, не обеспеченных необходимыми ресурсами
 - ▶ **Системность** планирования предполагает учет влияния на проект факторов его окружения, т.е. рассмотрение проекта как целостной системы с учетом взаимосвязей как внутри, так и вне его
 - ▶ **Гибкость** планирования предполагает способность системы прогнозировать и учитывать возможные изменения внешних факторов и их последствия
 - ▶ **Многофункциональность** планирования означает обязательное планирование по всем функциям управления проектом
 - ▶ **Оптимальность** планирования предполагает способность системы формировать не просто приемлемые (допустимые с точки зрения принятых ограничений) планы, а лучшие планы по выбранным критериям
 - ▶ **Непротиворечивость** планирования реализуется путем преемственности и взаимоувязанности всех плановых решений
 - ▶ **Непрерывность** планирования заключается в отслеживании, контроле а при необходимости – актуализации плановых решений
 - ▶ **Стабильность** планирования обеспечивается за счет неизменности основных целей проекта, его жизнеспособности
-

Основные процессы планирования

- ▶ планирование содержания проекта
- ▶ определение основных этапов реализации проекта, декомпозиция их на более мелкие и управляемые элементы
- ▶ формирование списка конкретных работ, которые обеспечивают достижение целей проекта
- ▶ установление последовательности работ с учетом их продолжительности, определение технологических зависимостей и ограничений на работы
- ▶ планирование ресурсов, определение того, какие ресурсы (люди, оборудование, материалы) и в каких количествах потребуются для выполнения работ проекта, уточнение сроков выполнения работ с учетом ограниченности ресурсов
- ▶ составление сметы, оценка стоимости ресурсов, необходимых для выполнения работ проекта
- ▶ составление бюджета, привязка сметных затрат к конкретным видам деятельности
- ▶ разработка плана проекта, сбор результатов остальных процессов планирования и их объединение в общий документ



Вспомогательные процессы планирования

- ▶ планирование качества: определение стандартов качества, соответствующих данному проекту, и поиск путей их достижения
- ▶ организационное планирование: распределение проектных ролей, ответственности и подчиненности
- ▶ подбор кадров: формирование команды проекта на всех стадиях жизненного цикла проекта
- ▶ планирование коммуникаций: определение кому и какая информация необходима, когда и как она им должна быть доставлена
- ▶ идентификация и оценка рисков: определение того, какой фактор неопределенности и в какой степени может повлиять на ход реализации проекта, разработка оптимистического и пессимистического сценария реализации проекта
- ▶ планирование поставок: определение того, каким образом, когда и с помощью кого закупать и поставлять

Планирование должно обеспечивать компромисс между требующимися характеристиками создаваемой системы или продукта и ограниченными ресурсами, необходимыми на их разработку и применение



Исходные данные для планирования

- ▶ договорные требования и обязательства
- ▶ информация о доступных ресурсах и ограничения на их использование (сроки, интенсивность, размещение и т. д.)
- ▶ оценочные и стоимостные модели
- ▶ стандарты (внутренние и внешние)
- ▶ результаты переговоров с заказчиком
- ▶ документация по аналогичным разработкам

Виды планов

- ▶ концептуальный план
- ▶ стратегический план
- ▶ тактические (детальные, оперативные) планы



SWOT-анализ (Strengths, Weaknesses, Opportunities and Threats — преимущества, слабые стороны, возможности, угрозы)

- ▶ каковы наши преимущества, как мы можем их реализовать?
- ▶ в чем наши слабые стороны, как мы можем уменьшить их влияние?
- ▶ какие существуют возможности, как мы можем извлечь выгоду из них?
- ▶ что могло бы воспрепятствовать угрозам?
- ▶ что мы могли бы сделать для каждого из обстоятельств, чтобы преодолеть или избежать возникновение проблемы?

Преимущества	Как их можно реализовать?	Слабые стороны	Как уменьшить их влияние?
Какие возможности предоставляет проект?	Как извлечь из них выгоду?	Угрозы: риски или иные обстоятельства, препятствующие успеху	Как можно минимизировать каждую из выявленных угроз?



Структурная декомпозиция работ (WBS, work breakdown structure)

Структурная декомпозиция работ представляет собой иерархический перечень рабочих действий, необходимых для завершения проекта. В этот перечень включаются действия, с помощью которых:

- ▶ осуществляется разработка ПО
- ▶ происходит управление проектом
- ▶ обеспечивается поддержка для всех процессов, выполняемых в ходе реализации проекта
- ▶ выполняются любые другие действия, необходимые для достижения целей проекта и удовлетворения потребностей пользователей, например, разработка документации и учебных материалов, закупка оборудования и инструментальных средств разработки и пр.

WBS – это *ориентированная на результат* иерархическая декомпозиция работ, выполняемых командой проекта для достижения целей проекта и получение необходимых результатов, с помощью которой *структурируется и определяется все его содержание*. Каждый следующий уровень иерархии содержит более детальное определение элементов проекта



Структурная декомпозиция работ (WBS, work breakdown structure)

- ▶ Структурная декомпозиция работ представляет собой инструмент, применяемый для документирования всех рабочих операций, которые должны быть выполнены при разработке и поставке ПО
- ▶ Структура WBS консолидирует информацию из различных источников, организуя ее с использованием единого формата, удобного для планирования и отслеживания хода реализации проекта
- ▶ На основе структуры WBS разрабатывается график реализации проекта

Структура WBS может быть создана на основе двух подходов к иерархии работ:

- ▶ представление продукта: отображает иерархические взаимосвязи между элементами продукта (подпрограммами, модулями, компонентами, подсистемами и т.д.)
- ▶ представление проекта: указывает на иерархические взаимосвязи среди рабочих действий (элементов процесса)



Для чего нужна схема WBS

- ▶ Схема WBS определяет работы, которые должны быть выполнены в ходе реализации проекта. Она гарантирует, что в состав проекта войдут все работы, которые позволят успешно его завершить
 - ▶ Схема WBS определяет сроки получения рабочих продуктов. Так как она разделена на низкоуровневые задачи, каждая из которых имеет дату начала и окончания, все участники проекта будут точно знать сроки завершения определенных работ
 - ▶ Если возникнет потребность добавить в существующий проект дополнительные функциональные возможности, это может быть отражено через изменение схемы WBS
 - ▶ Она дает возможность определить на соответствующем уровне детализации плана вехи (ключевые результаты), которые будут играть роль контрольных точек по проекту
 - ▶ На основе схемы WBS распределяется ответственность за достижение целей проекта между его исполнителями
 - ▶ Схема WBS обеспечивает членам команды понимание общих целей и задач по проекту
 - ▶ Схема WBS позволяет создать удобную, соответствующую целям проекта структуру отчетности
-



Структурная декомпозиция с точки зрения уровней управления

Уровни управления	Уровни иерархии	Наименование уровня иерархии
Организационно-экономический уровень	1	Общая программа (мега- или мультипроекта)
	2	Проект
	3	Подпроект
	4	Часть проекта
Технологический уровень	5	Комплекс работ
	6	Детальная работа
	7	Единичная работа

Уровень «Общая программа» позволяет определить и оценить место и роль данного проекта в окружении других проектов, объединенных общей программой

Уровни 2-4 характеризуют объектно-функциональную декомпозицию проекта и достаточны для всех верхних уровней руководства проектом (инвесторы, заказчик, топ-менеджмент организации)

Уровни 5-7 характеризуют декомпозицию, ориентированную на выполнение работы, они содержат информацию, необходимую для руководства работами на уровне исполнителей



Основания декомпозиции при построении схемы WBS

- ▶ компоненты продукта (объекта, услуги, направления деятельности), получаемого в результате реализации проекта
- ▶ процессные или функциональные элементы деятельности организации, реализующей проект
- ▶ этапы жизненного цикла проекта, основные фазы
- ▶ подразделения организационной структуры
- ▶ географическое размещение для пространственно распределенных проектов

Подход к декомпозиции с точки зрения каскадного подхода

ГОСТ 19.102-77 определяет следующие стадии разработки программной системы:

1. Техническое задание
3. Технический проект

2. Эскизный проект
4. Рабочий проект

5. Внедрение

Основные этапы построения WBS

- ▶ на основе информации, заложенной в концепции проекта, проводится последовательная декомпозиция работ проекта по заданным основаниям (критериям). Этот процесс продолжается до тех пор, пока все значимые работы, пакеты работ и отдельные задания не будут выделены и идентифицированы в такой степени и таким образом, чтобы они могли планироваться, для них можно было определять бюджет и составлять расписание, выполнять функции управления и контроля
- ▶ каждому элементу декомпозиции присваивается уникальный идентификатор, соответствующий уровню. Названия элементов на каждом уровне отражают критерии разбиения работ
- ▶ для каждой работы, пакета работ, части проекта, выделенных таким образом, определяются имеющие к ним отношение данные (продолжительность и трудозатраты, ответственные исполнители, бюджет и затраты, оборудование и материалы и т.д.)
- ▶ по каждой выделенной работе, пакету работ, части проекта проводится критический анализ с их исполнителями (членами команды проекта, менеджерами и другими участниками) для подтверждения правильности WBS



Возможные ошибки структуризации проекта

- ▶ пропуск стадии структуризации проекта и переход непосредственно к поиску и решению текущих, оперативных проблем проекта
- ▶ непонимание того, что схема WBS должна охватывать весь проект (обычно недостаточное внимание уделяется начальной и конечной фазам проекта, работе функциональных, обеспечивающих подразделений)
- ▶ излишняя или недостаточная детализация
- ▶ повторение элементов структуры



Традиционной (продуктовый) подход к построению схемы WBS при разработке ПО

Создание схемы **WBS** – это итерационный процесс, предполагающий постоянную переоценку иерархии поставляемых частей, действий и задач. Главный вопрос – это все ли части учтены, позволяют ли выделенные действия и задачи достичь цели проекта?

- ▶ Выделение основных поставляемых частей проекта
- ▶ Декомпозиция выделенных частей до уровня действий, дающих представление о технических требованиях каждой части
- ▶ Последующая декомпозиция каждого действия до уровня задач
- ▶ Назначение ответственных за выполнение каждой задачи
- ▶ Определение длительности каждой задачи либо на основе накопленных данных по реализованным проектам, либо с привлечением экспертного мнения членов команды разработчиков, ответственных за выполнение задач

Элементарным уровнем деления действий в схеме WBS рекомендуется считать уровень задач, выполняемых одним человеком за время от 1 до 10 дней



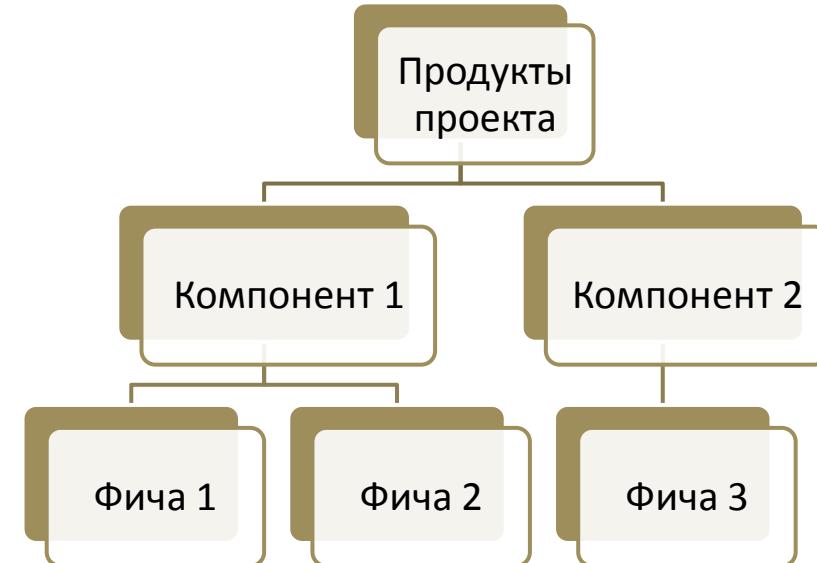
Традиционная декомпозиция работ, привязанная к структуре продукта



Проблемы традиционной схемы WBS при разработке ПО

- ▶ традиционные декомпозиции детализируются, планируются и финансируются либо слишком подробно, либо недостаточно подробно
- ▶ традиционные декомпозиции работ специфичны, для каждого проекта, поэтому сравнение разных проектов обычно оказывается затруднительным или невозможным

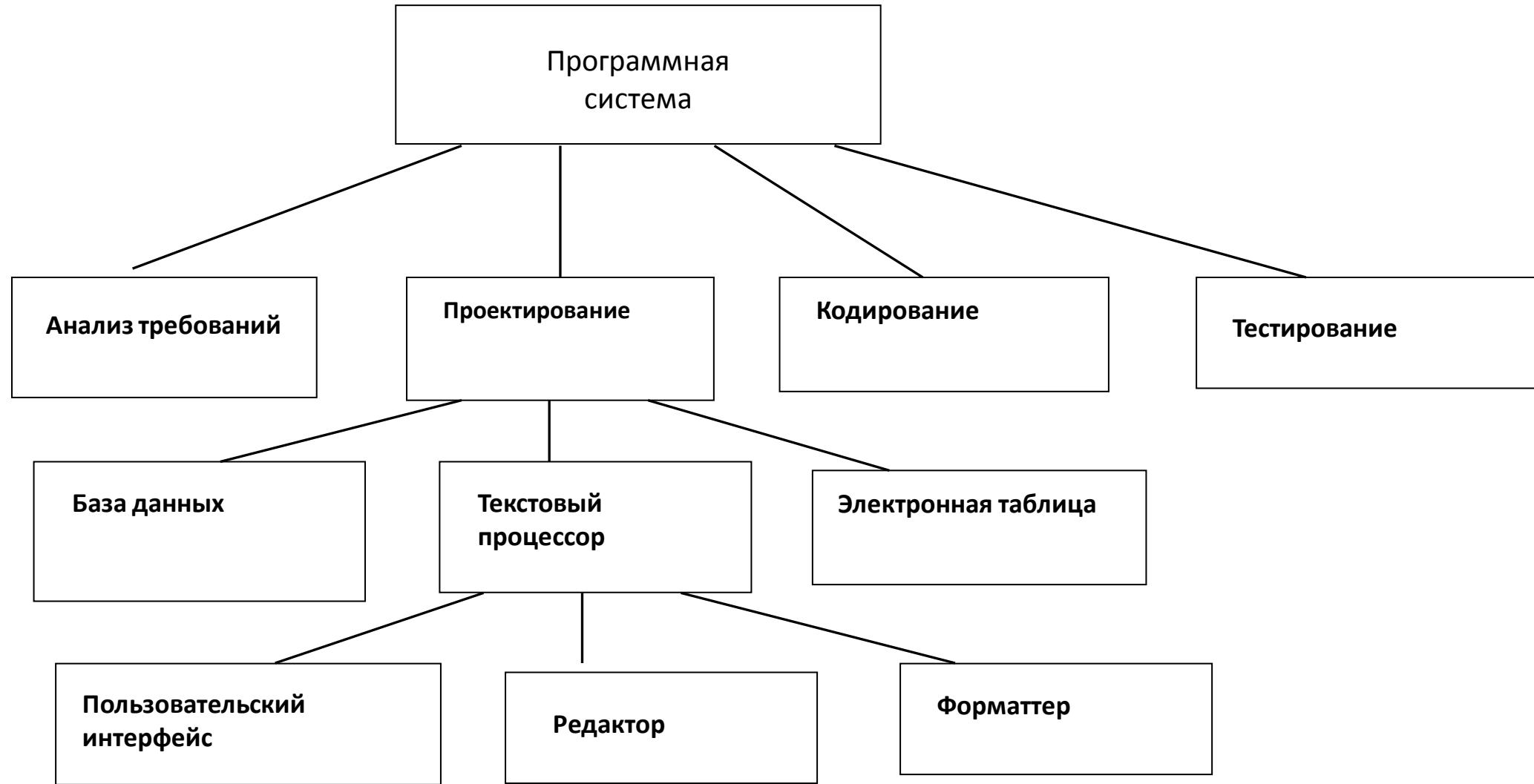
Выделение компонентов, составляющих программный продукт, это элемент высокоуровневого проектирования, которое должно быть выполнено на фазе планирования проекта, не дожидаясь проработки всех функциональных требований к разрабатываемому ПО



Примеры вопросов, на которые не дает ответов традиционная WBS

- ▶ Каково соотношение между производительными видами деятельности (требования, проектирование, реализация, оценка, внедрение) и непроизводительными (управление проектом, создание рабочей среды)?
 - ▶ Каков процент усилий, затраченных на доработки?
 - ▶ Каков процент затрат на основное оборудование и рабочее ПО (расходы на среду)?
 - ▶ Каково соотношение между тестированием и интеграцией?
 - ▶ Каковы затраты на версию N (являющиеся основанием для планирования затрат на версию N+1)?
-

Эволюционирующие декомпозиции работ



Рекомендации по построению эволюционирующих WBS

- ▶ Элементами WBS первого уровня являются рабочие процессы (управление проектом, создание рабочей среды, управление требованиями, проектирование, реализация, оценка и внедрение)
- ▶ Элементы второго уровня определяются для каждой стадии жизненного цикла
- ▶ Элементы третьего уровня определяются для выделения видов деятельности, в результате которых производятся рабочие продукты каждой стадии. Эти элементы могут либо образовывать самый нижний уровень в иерархии, который позволяет вычислить стоимость отдельного вида рабочих продуктов для данной стадии, либо разбиваться дальше на несколько задач более низких уровней, которые, взятые вместе, обеспечивают получение одного вида рабочих продуктов



Факторы, учитываемые при построении эволюционирующих WBS

- ▶ *Масштаб.* Более масштабные проекты будут иметь больше уровней и подструктур
 - ▶ *Организационная структура.* Проекты, где задействованы субподрядчики или участвует множество различных организаций, могут иметь ограничения, которые приведут к необходимости иного распределения работ
 - ▶ *Объем разработок на заказ.* В зависимости от характера проекта в рабочих процессах управления требованиями, проектирования и реализации внимание может уделяться разным аспектам
 - ▶ *Бизнес-контекст.* Проекты, выполняемые на контрактной основе, требуют более совершенного управления и оценки. Проекты, в которых разрабатываются коммерческие продукты для продажи широкому кругу потребителей, могут потребовать более совершенных структур для внедрения
 - ▶ *Предшествующий опыт.* Очень немногие проекты начинаются с чистого листа. Большинство из них разрабатывается либо как новые поколения существующих систем (с устоявшейся WBS), либо в контексте существующих организационных стандартов (с предопределенным построением WBS). Важно подстроиться под эти ограничения для гарантии, что новый проект сумеет воспользоваться имеющимся опытом и достигнутым уровнем производительности
-

Стандартные бюджеты WBS

Элемент WBS первого уровня	Стандартный бюджет
Управление проектом	10%
Создание рабочей среды	10%
Управление требованиями	10%
Проектирование	15%
Реализация	25%
Оценка	25%
Внедрение	5%
Итого	100%

Примечание:

в цифрах бюджета учтена стоимость различных категорий трудозатрат. Например, управление проектом, управление требованиями и проектирование — это элементы, где обычно используется персонал с более высокими должностями и с более высокой оплатой

в элемент создания рабочей среды включена также стоимость программной и аппаратной составляющих, необходимых для поддержки автоматизации процесса и команд разработчиков



Разработка схемы WBS методом сверху вниз

- ▶ Разработка схемы WBS начинается с самого верхнего элемента (поставляемого программного продукта), после чего идентифицируются большие рабочие продукты
- ▶ Руководитель программного проекта вырабатывает общую оценку размера проекта, процесса, среды, персонала и требуемого уровня качества
- ▶ Производится приблизительная оценка общих трудозатрат и сроков с использованием модели оценки стоимости
- ▶ Менеджер проекта детализирует эту приблизительную оценку трудозатрат на верхнем уровне WBS, используя рекомендации, аналогичные приведенным в таблице 1. На этом этапе детализируются также сроки путем установления основных контрольных точек, и распределяются необходимые трудозатраты в соответствии с квалификацией персонала
- ▶ В этот момент на руководителей отдельных направлений проекта возлагается ответственность за разбиение каждого из элементов WBS на элементы более низких уровней, учитывая их расположение на верхнем уровне, штатное расписание и даты основных контрольных точек в качестве ограничений



Планирование проекта методом снизу вверх

- ▶ Элементы WBS самого нижнего уровня прорабатываются в виде отдельных заданий, сроки и бюджеты для которых приблизительно оцениваются членами команды, ответственными за данный элемент WBS
- ▶ Приблизительные оценки суммируются и объединяются в бюджеты и контрольные точки более высоких уровней
- ▶ Производится сравнение с бюджетами и контрольными сроками, разработанными сверху вниз. Определяются самые значительные расхождения и делаются уточнения для того, чтобы достигнуть общего согласования между оценками, выполненными сверху вниз и снизу вверх



Создание структуры WBS при разработке ПО

Для создания структуры WBS при разработке ПО необходимо:

- ▶ Идентифицировать работы, связанные с созданием программного продукта, отделяя их от работ, связанных с аппаратным обеспечением и от рабочих процессов
- ▶ Найти структуру WBS для произвольной системы высшего уровня, отделяя ПО от других систем и компонентов
- ▶ Определить программную архитектуру WBS, идентифицируя все ее части и действия, требуемые при ее разработке
- ▶ Наполнить содержимым программную архитектуру WBS, идентифицируя все ее части и действия, требуемые при ее проектировании
- ▶ Определить категории затрат, связанных с ПО



Идентификация действий и задач

- ▶ Действие – элемент работы, выполняемый в ходе реализации проекта, характеризуемый ожидаемой длительностью и затратами, а также прогнозируемыми требованиями к ресурсам
- ▶ Действия могут подразделяться на задачи, которые рассматриваются как нижний уровень трудозатрат, понесенных при выполнении проекта
- ▶ Действия описываются «командным языком» с помощью глаголов и существительных, четко выражая определенную мысль
- ▶ Действия являются частями работы, которая может быть приемлемо выполнена с помощью одной единицы ресурса за относительно короткий промежуток времени

Примечание:

одна единица ресурса может означать один человек, одно структурное подразделение, один отдел и др. элемент организационной структуры

относительно короткий промежуток времени – день, неделя, месяц, либо другая единица времени, обладающая приемлемым уровнем детализации в масштабах области действия проекта, позволяющая адекватно оценить достижение целей проекта



Календарное планирование

Наименование работ (тема, работа, задача, задание)	Сроки выполнения начало/конец		Ответственный исполнитель и исполнители, роли	Требуемые ресурсы и сроки их предоставления план/факт	Примечания
	план	факт			
1	2	3	4	5	6

Календарный план — это поэтапно разбитая и упорядоченная по времени выполнения последовательность работ проекта



Построение рабочего графика проекта

Определение
этапов
выполнения
работ

Задание связей
между этапами

Оценка
ресурсов для
каждого этапа

Распределение
персонала по
этапам

Построение
графика



Задачи и виды деятельности, которые следует отражать в плане управления проектом (ISO 15504 «Оценка процессов разработки и поддержки ПО»)

- ▶ выбрать модель жизненного цикла, соответствующую назначению, функциям, масштабу и сложности проекта;
 - ▶ определить возможность достижения целей проекта в рамках существующих ресурсов и ограничений, включая возможные варианты достижения целей с учетом предполагаемых рисков;
 - ▶ определить работы, которые необходимо выполнить по проекту, количественно оценить их сложность, включая потребность в ресурсах, и принимая во внимание существующие риски и возможности разработать продукт с требуемым уровнем качества;
 - ▶ установить график (исполнители, сроки, ресурсы) выполнения проекта, основываясь на распределении работ, оценках и элементах инфраструктуры;
 - ▶ определить конкретные зоны ответственности для всех членов команды и обеспечить, чтобы обязанности были поняты и приняты, профинансираны и достижимы;
 - ▶ идентифицировать интерфейсы между элементами проекта, а также с другими проектами и организационными единицами системы;
 - ▶ определить инструментарий для обеспечения того, чтобы планы проекта были формально разработаны, реализованы, поддержаны и доступны лицам, вовлеченным в проект, обеспечить публикацию планов для специалистов, к которым они относятся;
 - ▶ регулярно оценивать степень выполнения проекта, принимать меры, для корректировки отклонений от плана и предотвращения повторения проблем, выявленных в проекте
-



Примерная структура плана реализации программного проекта

- ▶ *Введение.* Краткое описание целей проекта и проектных ограничений (бюджетных, временных и т.д.), которые важны для управления проектом
- ▶ *Организация выполнения проекта.* Описание способа подбора команды разработчиков и распределение обязанностей между членами команды
- ▶ *Анализ рисков.* Описание возможных проектных рисков, вероятности их проявления и стратегий, направленных на их уменьшение
- ▶ *Аппаратные и программные ресурсы, необходимые для реализации проекта.* Перечень аппаратных средств и программного обеспечения, необходимого для разработки программного продукта. Если аппаратные средства требуется закупать, приводится их стоимость совместно с графиком закупки и поставки
- ▶ *Разбиение работ на этапы.* Процесс реализации проекта разбивается на отдельные процессы, определяются этапы выполнения проекта, приводится описание результатов ("выходов") каждого этапа и контрольные точки
- ▶ *График работ.* В этом графике отображаются зависимости между отдельными процессами (этапами) разработки ПО, оценки времени их выполнения и распределение членов команды разработчиков по отдельным этапам
- ▶ *Механизмы мониторинга и контроля за ходом выполнения проекта.* Описываются предоставляемые руководителем отчеты о ходе выполнения работ, сроки их предоставления, а также механизмы мониторинга всего проекта



Сетевая модель проекта

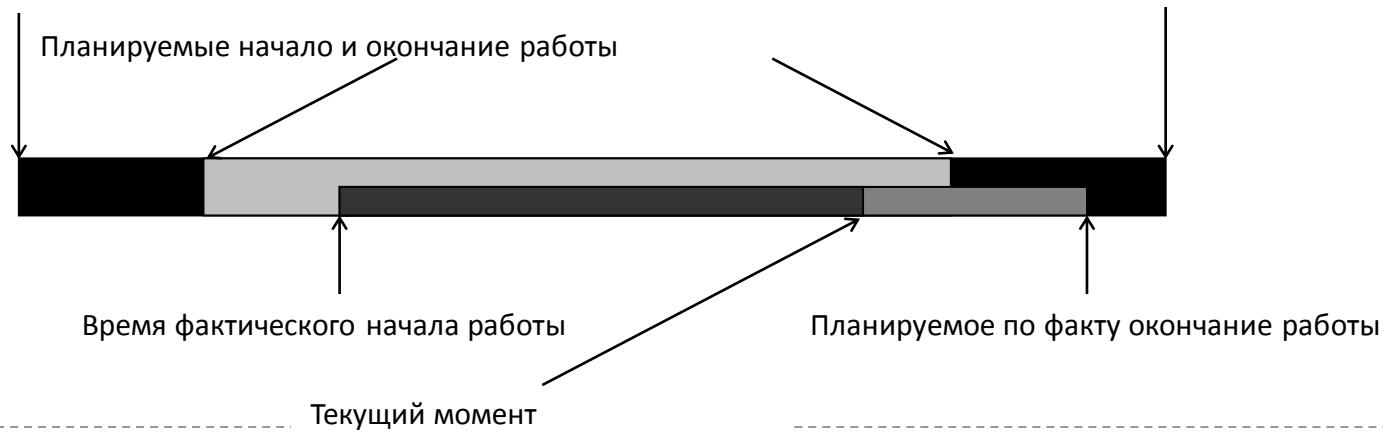
Процесс разработки сетевой модели включает в себя:

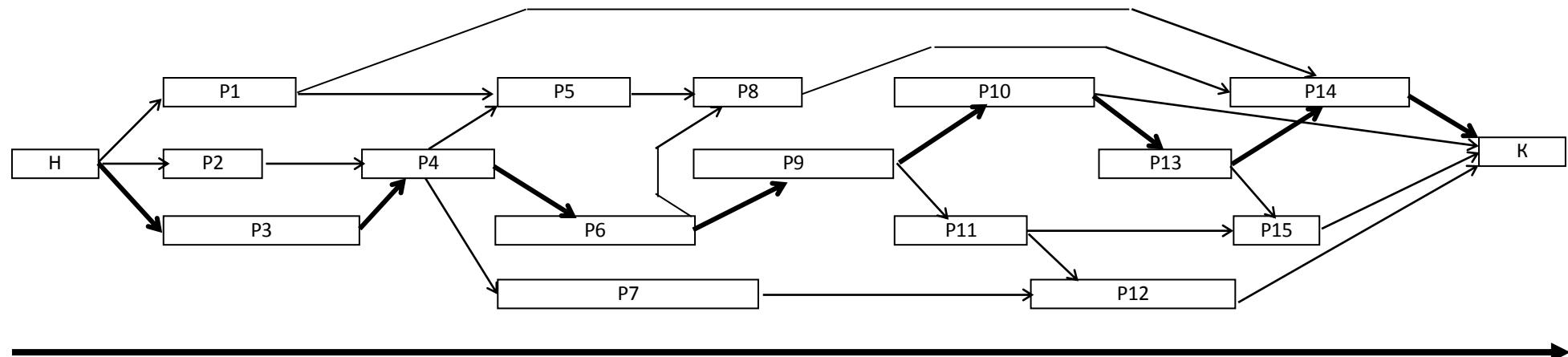
- ▶ определение списка работ проекта;
- ▶ оценку параметров работ;
- ▶ определение зависимостей между работами

Временные характеристики работы, учитываемые при построении сетевого графика

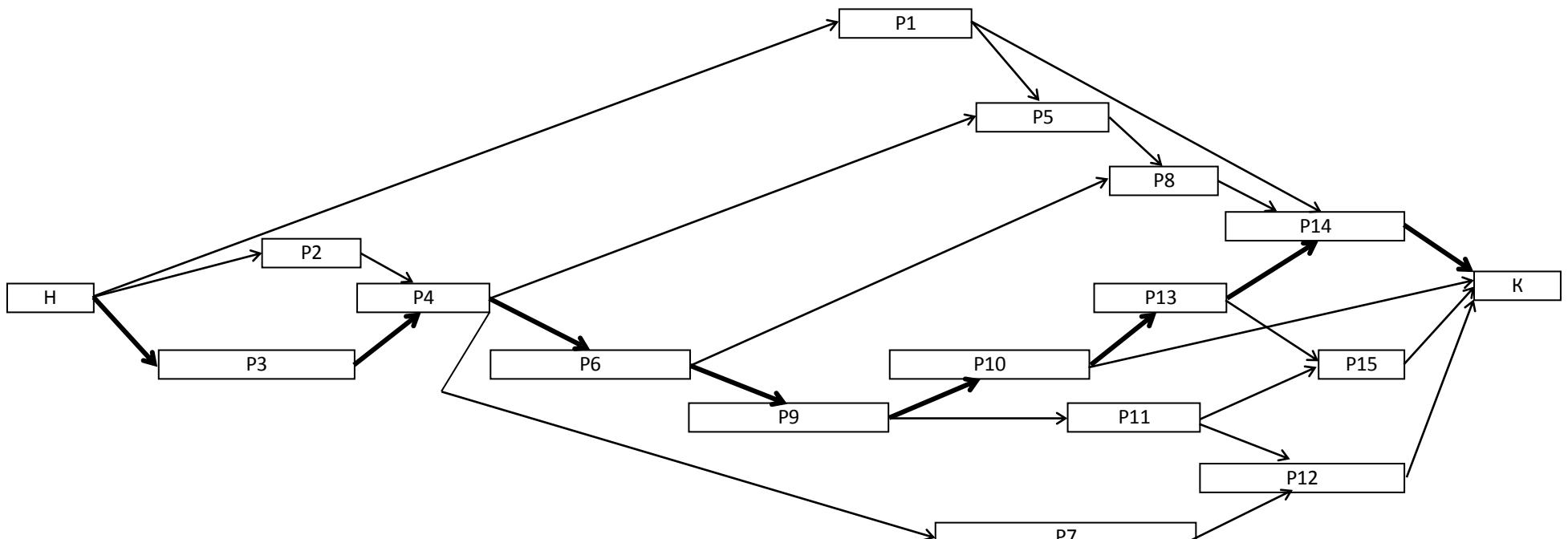
Время возможного начала работы

Время допустимого окончания работы





а) синхронизация начала работ



б) синхронизация окончания работ

Критический путь проекта

$$t(P3) + t(P4) + t(P6) + t(P9) + t(P10) + t(P13) + t(P14)$$

Максимальный по продолжительности полный путь в сети называется критическим. Длительность критического пути определяет наименьшую общую продолжительность работ по проекту в целом

При расчете критического пути учитываются следующие параметры:

- ▶ ранняя дата начала – это самая ранняя дата, с которой можно начать работу с учетом необходимости выполнения всех предшествующих задач с установленными для них временными ограничениями
- ▶ ранняя дата окончания – это самая ранняя дата, в которую можно закончить работу с учетом ее продолжительности и необходимости выполнения всех предшествующих работ, с установленными для них временными ограничениями
- ▶ поздняя дата начала – это самая поздняя дата, в которую возможно начать работу без изменения продолжительности критического пути и даты завершения проекта
- ▶ поздняя дата окончания – это самая поздняя дата, в которую возможно завершить работу без изменения продолжительности критического пути и даты завершения проекта

Практическая рекомендация: на критическом пути должны стоять работы с нежесткими связями, которые всегда можно перепланировать, если возникает угроза срыва сроков

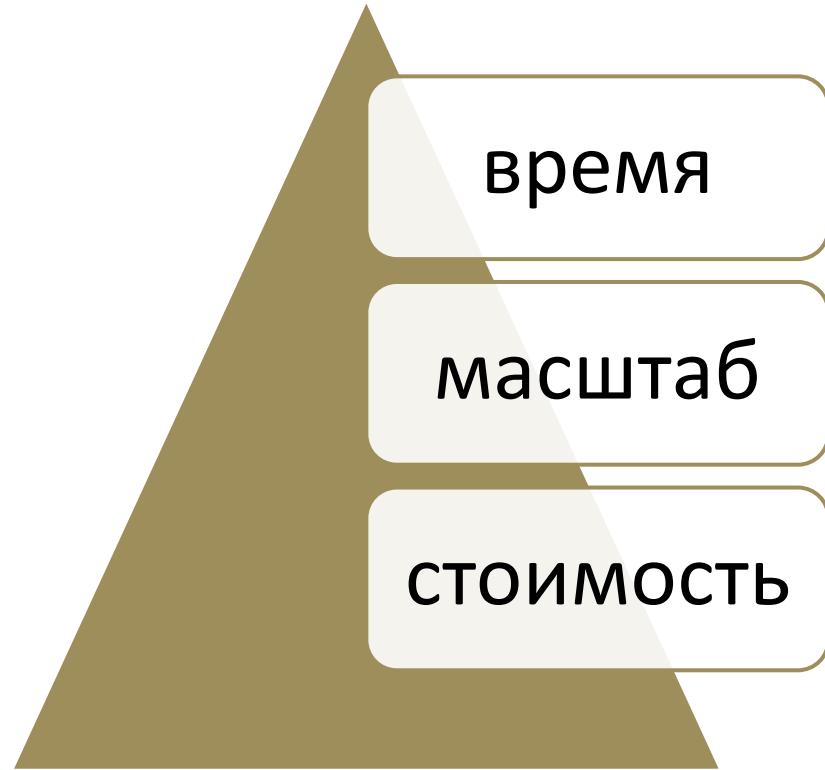


Данные, необходимые для расчета календарного графика проекта

- ▶ набор работ
- ▶ зависимости между работами
- ▶ оценки продолжительности каждой работы
- ▶ календарь рабочего времени проекта (в наиболее общем случае возможно задание собственного календаря для каждой работы)
- ▶ календари ресурсов
- ▶ ограничения на сроки начала и окончания отдельных работ или этапов
- ▶ календарная дата начала проекта



Треугольник проекта



Треугольник работы



$$\text{Трудозатраты} = \text{Длительность} * \text{Количество ресурсов}$$



Типы работ в проекте

- ▶ *работа с фиксированной длительностью* имеет определенную продолжительность, которая не зависит от количества назначенных ей ресурсов: выполнение такой работы нельзя ускорить, назначив на нее больше исполнителей. Если такая задача оказывается одной из критических в проекте и проект отстает от графика, ликвидировать отставание можно, только добавляя ресурсы для ее предшественников
 - ▶ *работа с фиксированными трудозатратами* имеет длительность, зависящую от количества назначенных исполнителей (ресурсов). Для таких работ, если добавить или удалить единицы ресурсов (или изменить процент их доступности в проекте) будет пересчитана продолжительность. Т.е. для таких задач увеличение числа исполнителей приведет к сокращению времени выполнения работы
 - ▶ *работа с фиксированным количеством единиц ресурсов*: при первом выделении ресурсов вычисляется объем работ (т.е. трудозатраты) путем умножения продолжительности работы на число единиц ресурсов. После этого число единиц объявляется постоянной величиной и если для такой задачи попытаться добавить или удалить исполнителей (или изменить процент их доступности), то будет пересчитана продолжительность. Т.е. по мере реализации проекта можно ликвидировать отставание таких задач, вручную добавляя единицы ресурсов, так как при добавлении ресурсов за то же самое время они могут выполнить больший объем работы
-

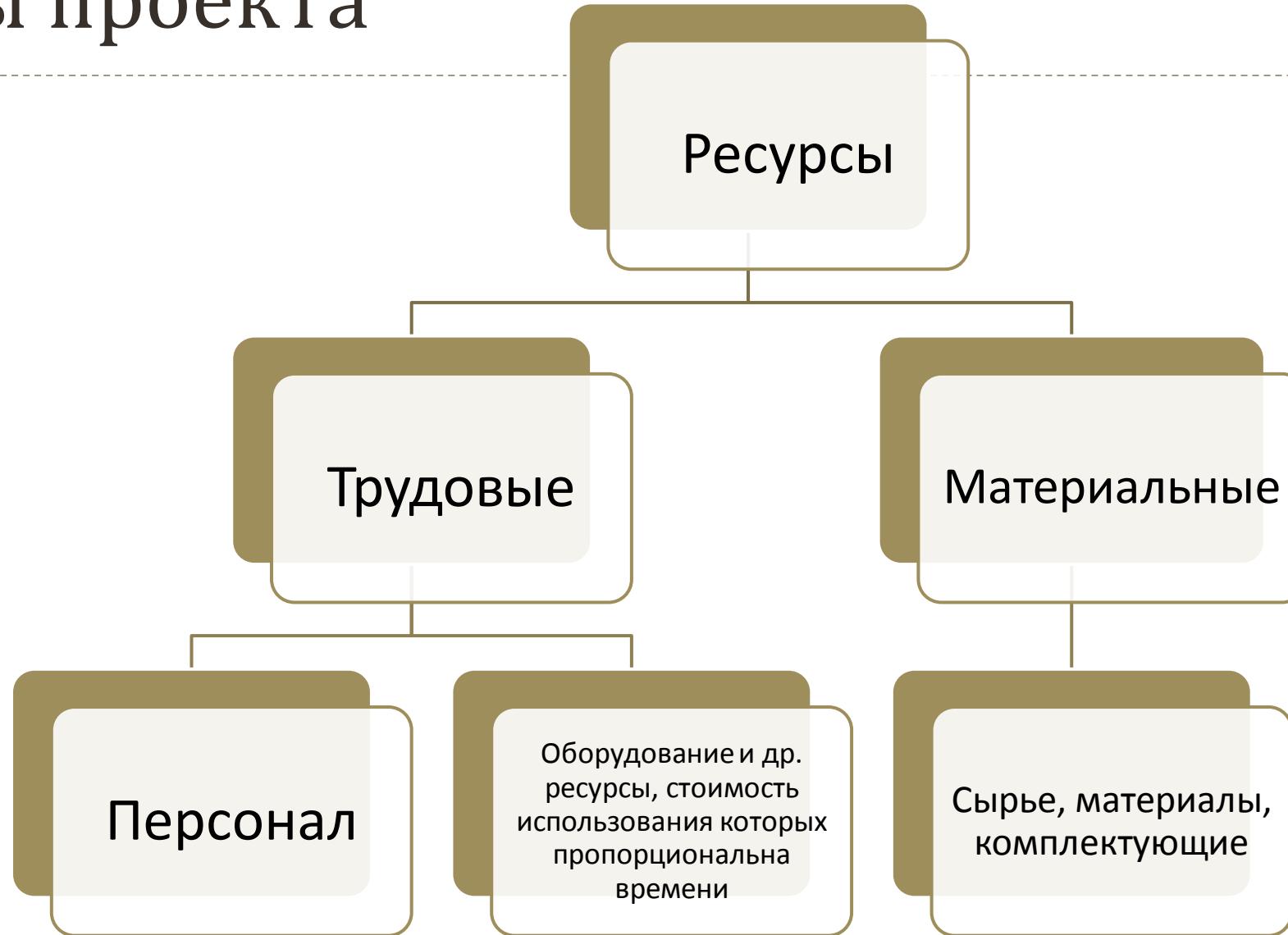


Типы зависимостей между работами

- ▶ «начало после окончания». Это стандартная последовательность, при которой предшествующая работа должна завершиться до начала последующей
- ▶ «начало после начала». Это наиболее общая последовательность при моделировании работ, которые должны выполняться одновременно. В этом случае не требуется завершения предшествующей работы до начала последующей. Для ее начала необходимо, чтобы предшествующая работа только началась
- ▶ «окончание после окончания». Этот тип зависимости также используется для моделирования параллельных работ. В этом случае окончание последующей работы контролируется окончанием работы предшественницы
- ▶ «окончание после начала». Последующая задача заканчивается, когда начинается предшествующая . Этот тип зависимости используется довольно редко и применяется прежде всего для работ, выполняемых вахтовым методом



Ресурсы проекта



Типы ресурсов в проекте

- ▶ *Невоспроизводимые*, складируемые, накапливаемые ресурсы в процессе выполнения работ расходуются полностью, не допуская повторного использования. Не использованные в данный отрезок времени, они могут использоваться в дальнейшем. Иными словами, такие ресурсы можно накапливать с последующим расходованием запасов. Поэтому их часто называют ресурсами типа «энергия». Примерами таких ресурсов являются топливо, предметы труда, средства труда однократного применения, а также финансовые средства
- ▶ *Воспроизводимые*, нескладируемые, ненакапливаемые ресурсы в ходе работы сохраняют свою натурально-вещественную форму и по мере высвобождения могут использоваться на других работах. Если эти ресурсы простаивают, то их неиспользованная способность к функционированию в данный отрезок времени не компенсируется в будущем, т.е. они не накапливаются. Поэтому ресурсы второго типа называют еще ресурсами типа «мощности». Примерами ресурсов типа «мощности» являются люди и средства труда многократного использования (машины, механизмы, станки и т.п.)



Этапы ресурсного планирования

- ▶ определение типов ресурсов, необходимых для реализации требуется проекта
- ▶ определение количества каждого типа ресурсов
- ▶ определение источника поступления ресурсов каждого типа
- ▶ назначение ресурсов задачам
- ▶ анализ расписания и разрешение возникших противоречий между требуемым количеством ресурсов и количеством, имеющимся в наличии

Алгоритмы устранения перегрузки ресурсов

- ▶ изменить календарь работы ресурса;
 - ▶ назначить ресурс на неполный рабочий день;
 - ▶ изменить профиль назначения ресурса;
 - ▶ добавить ресурсу время задержки;
 - ▶ разбить задачу на этапы и перекрыть по времени их выполнение
-



Спасибо за внимание!

Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана
Каф. ИУ-7

baryshnikovam@mail.ru

Лекция 6

Экономическая модель разработки ПО. Оценка технико-экономических показателей проекта. Оценка размера программного продукта в строках программного кода. Модели СОСОМО и СОСОМО II. Влияние эффектов повторного использования кода на размер ПО

Несоответствие производительности изначально предполагаемым показателям может иметь две следующие причины: плохая работа или некорректная оценка. В мире программного обеспечения можно получить множество свидетельств плохо выполненной оценки, однако практически невозможно доказать, что персонал не трудился усердно над разрешением проблемы, либо не является в достаточной степени компетентным

Том Де Марко



Технико-экономическое обоснование программного проекта

представляет собой процедуру оценивания трудовых, временных и финансовых ресурсов по созданию программного продукта, соответствующего требованиям заказчика

Объем требуемых ресурсов зависит от:

- ▶ совокупности бизнес-процессов, описывающих предметную область, и их приоритетов для заказчиков
- ▶ требований к функциональной полноте и качеству реализации каждого бизнес-процесса



Две части жизненного цикла программных средств

- ▶ Первая часть – включает в себя работы по системному анализу, проектированию, разработке, тестированию и испытаниям базовой версии программного продукта
- ▶ Вторая часть - отражает эксплуатацию, сопровождение, модификацию, управление конфигурацией и перенос ПС на иные платформы

Сложные программные продукты являются одними из наиболее сложных объектов, создаваемых человеком, и в процессе их производства творчество специалистов в контексте поиска новых методов, альтернативных решений и способов реализации заданных требований, а также формирование и декомпозиция этих требований составляют значительную часть всех трудозатрат



Исходные данные, используемые для прогнозирования и планирования

- ▶ функции и номенклатура характеристик самого прогнозируемого объекта или процесса
- ▶ характеристики прототипов и пилотных проектов, в некоторой степени подобных планируемому объекту, которые уже завершились и в отношении которых известны необходимые экономические характеристики и можно оценить качество процессов планирования и прогнозирования

Типовое распределение стоимости между основными этапами жизненного цикла (без сопровождения)

- ▶ 25% - проектирование – разработка и верификация проекта
- ▶ 15% - спецификации – формулировка требований и условий разработки
- ▶ 20% - разработка – кодирование и тестирование компонент
- ▶ 40% - интеграция и тестирование



Причины для сравнения реальных данных с прогнозируемыми оценками характеристик проекта

- ▶ несовершенство исходных данных при оценивании технико-экономических показателей программных проектов вызывает необходимость периодического пересмотра прогнозных оценок с учетом новой информации, чтобы обеспечить более реальную основу для дальнейшего управления проектом
- ▶ вследствие несовершенства методов оценивания технико-экономических показателей программных проектов следует сравнивать прогнозные оценки этих показателей с действительными значениями, формирующимиися в ходе реализации проекта, и использовать эти результаты для улучшения самих методов оценивания
- ▶ программные проекты имеют тенденцию к изменению характеристик и экономических факторов, поэтому для их успеха необходимо идентифицировать эти изменения и выполнять реалистичное обновление оценок затрат



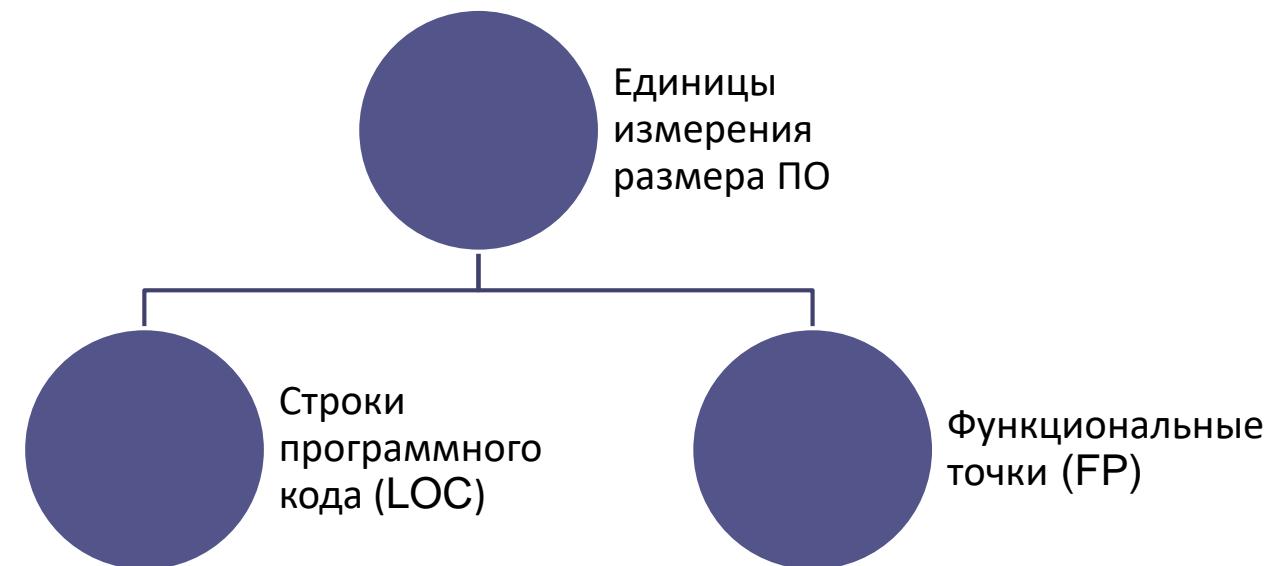
Факторы, повышающие точность оценок

- ▶ цели оценивания технико-экономических показателей должны быть согласованы с потребностями в информации, способствующей принятию решений на соответствующем этапе программного проекта
- ▶ достоверность оценок должна быть сбалансирована для различных компонентов системы и величина уровня неопределенности для каждого компонента должна быть примерно одинаковой, если в процессе принятия решения все компоненты имеют одинаковый вес
- ▶ следует возвращаться к предшествующим целям оценивания технико-экономических показателей и изменять их, когда это необходимо для ответственных бюджетных решений, принимаемых на ранних этапах и влияющих на следующие этапы



Основные параметры оценки при создании ПП

- ▶ сложность (размеры)
- ▶ трудозатраты на разработку
- ▶ длительность разработки в целом и ее отдельных этапов
- ▶ численность и квалификация специалистов, привлекаемых к созданию ПП



Проблемы использования LOC в качестве единицы измерения размера программного продукта

- ▶ число строк исходного кода зависит от уровня мастерства программиста. Фактически, чем выше мастерство программиста, тем меньшим количеством строк кода ему удастся обойтись для реализации определенной функциональной возможности (или функциональности) ПС
- ▶ высокоуровневые языки или языки визуального программирования требуют гораздо меньшего числа строк кода для отражения одной и той же функциональности, чем, например, язык Ассемблера или С. Очевидно, что существует обратная взаимосвязь между уровнем языка и производительностью труда (количеством строк кода в день) программиста
- ▶ фактическое число строк программного кода остается неизвестным до тех пор, пока проект не будет почти завершен. Поэтому LOC сложно использовать для предварительной оценки трудозатрат на разработку и построения плана-графика проекта
- ▶ в программистском сообществе не достигнуто соглашения о методе подсчета строк кода. Языковые конструкции, используемые, например, в Visual C++, Ассемблере, Коболе или SQL существенно различаются. Метод же остается общим для любых приложений, в том числе использующих комбинацию различных языков
- ▶ заказчику сложно понять, каково соотношение указанных им функциональных и нефункциональных (технических) требований к ПС и объемов программистской работы



Рекомендации по повышению достоверности LOC - оценок

- ▶ убедитесь, что каждая учитываемая строка исходного кода содержит лишь один оператор. Если в одной строке содержатся два выполняемых оператора, разделенных точкой с запятой, то они должны учитываться как две строки. Если же один оператор разбит на несколько «физических» строк, он будет учитываться как одна строка. В языках программирования допускаются различные правила кодирования, но обычно проще определять в строке один оператор, обрабатываемый компилятором или интерпретатором
- ▶ учитывайте все выполняемые операторы. Конечный пользователь может не иметь возможности практически использовать каждый оператор, но все операторы должны поддерживаться данным продуктом, в том числе и утилитами
- ▶ определения данных учитывайте лишь один раз
- ▶ не учитывайте строки, содержащие комментарии
- ▶ не учитывайте отладочный код либо другой временный код (пробное ПО, средства тестирования и пр.)
- ▶ учитывайте каждую инициализацию, вызов или включение макроса (директивы компилятора) в качестве части исходного кода, в которой осуществляется какое-либо действие. Не учитывайте повторно используемые операторы



Пример оценки показателя LOC с помощью экспертных оценок

Некоторый объект, отраженный на структуре WBS, по мнению экспертов может занимать от 200 до 400 строк кода, но скорее всего, его размер ближе к 250 строкам

Используя метод PERT, размер объекта можно оценить в 266 строк:

$$(200+(250*4)+400)/6 = 266 \text{ LOC}$$

Оценка количества LOC по аналогии

Например, у нас имеется уже готовый модуль А, размер которого составляет 2345 LOC. Мы хотим создать новый модуль А', который будет во многом схож с модулем А, но в него будут добавлены некоторые дополнительные свойства. Кроме того, мы знаем как сделать программный код более компактным. В результате этого размер модуля А' может быть оценен в 3000 LOC



Использование языка ассемблера для сравнительного анализа различных проектов

Язык программирования	Basic Assembler
Ассемблер	1
С	2,5
Кобол	3
Фортран	3
Паскаль	3,5
C++	6
Java	6
Ada 95	6,5
Access	8,5
Delphi Pascal	11
CORBA	16

Пусть решается задача по переводу некоторой операционной системы, написанной на языке С и содержащей 50000 строк кода на язык C++

Операционная система, содержащая 50000 строк кода на языке С, эквивалентна 125000 строкам кода на языке ассемблера, которые, будучи переписаны на C++, составят 20833 LOC

Примечание:

$$50000 * 2,5 = 125000$$

$$125000 / 6 = 20833$$



Преимущества использования LOC в качестве единицы измерения

- ▶ Эти единицы широко распространены и могут адаптироваться
- ▶ Они позволяют проводить сопоставление методов измерения размера и производительности в различных группах разработчиков
- ▶ Они непосредственно связаны с конечным продуктом
- ▶ Единицы LOC могут быть оценены еще до завершения проекта
- ▶ Оценка размеров ПО производится с учетом точки зрения разработчиков
- ▶ Действия по непрерывному улучшению базируются на количественных оценках. При этом спрогнозированный размер может быть легко сопоставлен с реальным размером на этапе постпроектного анализа. Это позволяет экспертам накапливать опыт и улучшать сами методы оценки
- ▶ Знание размера программного продукта в LOC – единицах позволяет применять большинство существующих методов оценки технико-экономических показателей проекта (таких как трудозатраты, длительность проекта, его стоимость и др.)



Недостатки, связанные с применением LOC – оценок

- ▶ Данные единицы измерения сложно применять на ранних стадиях жизненного цикла, когда высок уровень неопределенности
- ▶ Исходные инструкции могут различаться в зависимости от языков программирования, методов проектирования, стиля и способностей программистов
- ▶ Применение методов оценки с помощью количества строк не регламентируется промышленными стандартами, например ISO
- ▶ Разработка ПО может быть связана с большими затратами, которые напрямую не зависят от размеров программного кода (это затраты, связанные с разработкой спецификации требований, подготовкой пользовательской документации и пр., которые не включены в прямые затраты на кодирование)
- ▶ При подсчете LOC – единиц следует различать автоматически генерированный код и код, написанный вручную, что сильно затрудняет применение автоматических методов подсчета
- ▶ Генераторы кода зачастую провоцируют его избыточный объем, что может привести к значительным погрешностям в оценке размера ПП
- ▶ Единственным способом получения LOC – оценки является сравнение с аналогичными разработками или экспертные мнения, а эти методы изначально не относятся к числу точных



Способ учета качества программного кода

Программисты могут быть незаслуженно премированы за достижение высоких показателей LOC, если служба менеджмента посчитает это высоким признаком продуктивности

При этом показатели LOC не могут осуществляться для сравнения производительности различных команд разработчиков (либо для нормирования труда) в случае, если использовались разные платформы или типы языков программирования

Количество дефектов / количество строк кода

Софтверные организации склонны вознаграждать программистов, которые: а) пишут много кода; б) исправляют много ошибок. Соответственно, наилучший способ отличиться в таких условиях – это создать большое количество некачественного кода, а потом героически устранять в нем собственные же промахи

Джоэль Спольски (Joel Spolsky) - руководитель компании Fog Creek Software



Экономическая модель разработки ПО

Трудозатраты= (Персонал)(Среда)(Качество)(Размер^{Процесс})

Размер – размер конечного продукта (для компонентов, написанных вручную), который обычно измеряется числом строк исходного кода или количеством функциональных точек, необходимых для реализации данной функциональности. В это понятие также должны входить и другие создаваемые материалы, такие как документация, совокупность тестовых данных и обучающие материалы

Персонал – возможности персонала, участвующего в разработке ПО, в особенности его профессиональный опыт и знание предметной области проекта. Источниками сложностей могут быть требуемая надежность программного обеспечения, ограничения на производительность и хранение, требуемое повторное использование программных компонентов, а так же опыт работы программистов с данной средой программирования

Среда – состоит из инструментов и методов, используемых для эффективной разработки ПО и автоматизации процесса. Т.е. фактически, это приобретенная или потерянная эффективность вследствие уровня автоматизации процесса (больший уровень автоматизации приводит к уменьшению усилий и повышению эффективности)

Качество – требуемое качество продукта, что включает в себя его функциональные возможности, производительность, надежность и адаптируемость

Процесс – особенности процесса, используемого для получения конечного продукта, в частности, его способность избегать непроизводительных видов деятельности: переделок, бюрократических проволочек, затрат на взаимодействие



Алгоритм оценки затрат на разработку ПО

- ▶ оценка размера разрабатываемого продукта
- ▶ оценка трудозатрат (трудоемкости) в человеко-месяцах или человеко-часах
- ▶ оценка продолжительности проекта в календарных месяцах
- ▶ оценка стоимости проекта

Основным интегральным экономическим показателем каждого программного проекта, отражающим суммарные затраты интеллектуального труда специалистов на производство программного продукта, является трудозатраты (трудоемкость)

Полный анализ и оптимизацию суммарных затрат на проект целесообразно проводить на всем протяжении жизненного цикла, при этом в ряде случаев очень важно учитывать в том числе затраты на сопровождение и эксплуатацию программного продукта

Эти виды затрат характеризуются значительной неопределенностью из-за сложности прогнозирования длительности жизненного цикла, требований качества, степени модификации программ и затрат на сопровождение



Модель оценки стоимости COCOMO (COnstructive COst MOdel — конструктивная модель стоимости)

Трудозатраты = C1 * EAF * (Размер)^{p1}

Время = C2 * (Трудозатраты)^{p2}

Трудозатраты (работа) — количество человеко-месяцев;

C1 — масштабирующий коэффициент

EAF — уточняющий фактор, характеризующий предметную область, персонал, среду и инструментарий, используемый для создания рабочих продуктов процесса

Размер — размер конечного продукта (кода, созданного человеком), измеряемый в исходных инструкциях (DSI, delivered source instructions), которые необходимы для реализации требуемой функциональной возможности

P1 — показатель степени, характеризующий экономию при больших масштабах, присущую тому процессу, который используется для создания конечного продукта; в частности, способность процесса избегать непроизводительных видов деятельности (доработок, бюрократических проволочек, накладных расходов на взаимодействие)

Время — общее количество месяцев

C2 — масштабирующий коэффициент для сроков исполнения

P2 — показатель степени, который характеризует инерцию и распараллеливание, присущие управлению разработкой ПО



Допущения модели СОСМО

- ▶ Исходные инструкции конечного продукта включают в себя все (кроме комментариев) строки кода, обрабатываемого компьютером
- ▶ Начало жизненного цикла проекта совпадает с началом разработки продукта, окончание — совпадает с окончанием приемочного тестирования, завершающего стадию интеграции и тестирования
- ▶ Работа и время, затрачиваемые на анализ требований, оцениваются отдельно, как дополнительный процент от разработки в целом
- ▶ Виды деятельности включают в себя только работы, направленные непосредственно на выполнение проекта
- ▶ Человеко-месяц состоит из 152 часов
- ▶ Проект управляется надлежащим образом, в нем используются стабильные требования



Режимы модели СОСМО

Название режима	Размер проекта	Описание	Среда разработки
Обычный	До 50 KLOC	Некрупный проект разрабатывается небольшой командой, для которой нехарактерны нововведения, разработчики знакомы с инструментами и языком программирования	Стабильная
Промежуточный	50 – 500 KLOC	Относительно небольшая команда занимается проектом среднего размера, в процессе разработки необходимы определенные инновации	Среда характеризуется незначительной нестабильностью
Встроенный	Более 500 KLOC	Большая команда разработчиков трудится над крупным проектом, необходим значительный объем инноваций	Среда состоит из множества нестабильных элементов



Формулы для оценки основных работ и сроков

Обычный вариант

Трудозатраты = $3,2 * \text{EAF} * (\text{Размер})^{1,05}$

Время (в месяцах) = $2,5 * (\text{Трудозатраты})^{0,38}$

Трудозатраты (работа) — количество человеко-месяцев

EAF — результат учета 15 уточняющих факторов (см. таблицу)

Размер — число исходных инструкций конечного продукта (измеряемое в тысячах строк кода KLOC)

Промежуточный вариант

Трудозатраты = $3,0 * \text{EAF} * (\text{Размер})^{1,12}$

Время (в месяцах) = $2,5 * (\text{Трудозатраты})^{0,35}$

Встроенный вариант

Трудозатраты = $2,8 * \text{EAF} * (\text{Размер})^{1,2}$

Время (в месяцах) = $2,5 * (\text{Трудозатраты})^{0,32}$



Значение драйверов затрат в модели СОСМО

Идентификатор	Уточняющий фактор работ	Диапазон изменения параметра	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий
Атрибуты программного продукта							
RELY	Требуемая надежность	0,75-1,40	0,75	0,86	1,0	1,15	1,4
DATA	Размер базы данных	0,94-1,16		0,94	1,0	1,08	1,16
CPLX	Сложность продукта	0,70-1,65	0,7	0,85	1,0	1,15	1,3
Атрибуты компьютера							
TIME	Ограничение времени выполнения	1,00-1,66			1,0	1,11	1,50,
STOR	Ограничение объема основной памяти	1,00-1,56			1,0	1,06	1,21
VIRT	Изменчивость виртуальной машины	0,87-1,30		0,87	1,0	1,15	1,30
TURN	Время реакции компьютера	0,87-1,15		0,87	1,0	1,07	1,15
Атрибуты персонала							
ACAP	Способности аналитика	1,46-0,71	1,46	1,19	1,0	0,86	0,71
AEXP	Знание приложений	1,29-0,82	1,29	1,15,	1,0	0,91	0,82
PCAP	Способности программиста	1,42-0,70	1,42	1,17	1,00	0,86	0,7
VEXP	Знание виртуальной машины	1,21-0,90	1,21	1,1	1,0	0,9	
LEXP	Знание языка программирования	1,14-0,95	1,14	1,07	1,0	0,95	
Атрибуты проекта							
MODP	Использование современных методов	1,24-0,82	1,24	1,1	1,0	0,91	0,82
TOOL	Использование программных инструментов	1,24-0,83	1,24	1,1	1,0	0,91	0,82
SCED	Требуемые сроки разработки	1,23-1,10	1,23	1,08	1,0	1,04	1,1



Распределение работ и времени по стадиям жизненного цикла

Вид деятельности	Трудозатраты (%)	Время (%)
Планирование и определение требований	(+8)	(+36)
Проектирование продукта	18	36
Детальное проектирование	25	18
Кодирование и тестирование отдельных модулей	26	18
Интеграция и тестирование	31	28

Примечание: В основе распределения трудозатрат и времени лежит каскадная модель жизненного цикла



Декомпозиция работ по созданию ПО

<i>Вид деятельности</i>	<i>Бюджет (%)</i>
Анализ требований	4
Проектирование продукта	12
Программирование	44
Тестирование	6
Верификация и аттестация	14
Канцелярия проекта	7
Управление конфигурацией и обеспечение качества	7
Создание руководств	6
Итого	100



Пример использования модели COCOMO

По контракту с государственной организацией разрабатывается большая, критически важная система (например, для управления электростанцией). Объем программного кода был предварительно оценен в 100 000 строк (100 KLOC). Используемая технология является новой для разработчиков.

Произвести оценку параметров по методике COCOMO

Все драйверы затрат номинальные, кроме:

Фактор, влияющий на стоимость	Идентификатор	Значение	Значение параметра
Использование программных инструментов	TOOL	Высокое	0,88
Знание приложений	AEXP	Низкое	1,1
Требуемая надежность	RELY	Высокое	1,15
Сложность продукта	CPLX	Высокое	1,15
EAF			1,28



Расчеты по методике СОСОМО

- ▶ Трудозатраты = $2,8 * \text{EAF} * (\text{KDSI})^{1,2} = 2,8 * 1,28 * (100)^{1,2} = 900$ человеко-месяцев на разработку + 72 человека-месяца на планирование, определение требований = 972 человека-месяца
- ▶ Время = $2,5 * (\text{Трудозатраты})^{0,32} = 2,5 * (900)^{0,32} = 22$ месяца на разработку + 8 месяцев на планирование, определение требований = 30 месяцев

Учитывая критическую важность и сложность системы для расчетов использовался встроенный вариант



Распределение работ по этапам жизненного цикла

Вид деятельности	Бюджет (%)	Человеко-месяцы
Анализ требований	4	39
Проектирование продукта	12	117
Программирование	44	428
Тестирование	6	58
Верификация и аттестация	14	136
Канцелярия проекта	7	68
Управление конфигурацией и обеспечение качества	7	68
Создание руководств	6	58
Итого	100	972



Достоинства модели СОСМО

- ▶ Метод является достаточно универсальным и может поддерживать различные режимы и уровни программных разработок
- ▶ При расчетах используются множители и показатели степени, полученные на основе анализа данных большого количества практически реализованных проектов
- ▶ Предложенные драйверы затрат хорошо подгоняются под специфику конкретной организации
- ▶ Точность оценок повышается по мере накопления в организации опыта применения модели
- ▶ Метод снабжен обширной документацией и прост в применении



Недостатки модели СОСМО

- ▶ Все уровни зависят от оценки размера – точность оценки размера оказывает влияние на точность оценки трудозатрат, времени разработки, подбор персонала и оценку производительности
- ▶ Метод основан на каскадной модели жизненного цикла и прежде всего не учитывает изменяемость требований
- ▶ Слишком поверхностное вниманиеделено вопросам обеспечения безопасности и надежности
- ▶ Модель не учитывает возможности повторного использования кода, итерационные возвраты по этапам жизненного цикла, объектно-ориентированные технологии разработки ПО



Концепция повторного использования

- ▶ Многие программы происходят от предыдущих версий этих же программ
- ▶ В результате может быть достигнута экономия средств и/или времени, а также повышен уровень качества
- ▶ Повторному использованию подлежат: программный код, тестовый код, тестовые процедуры, документация, дизайн, спецификации требований и др.
- ▶ Код, повторно используемый в полном объеме, имеет идентичную документацию, идентичные тестовые процедуры и сам тестовый код и только одну копию, поддерживаемую системой управления конфигураций



Повторное использование кода

- ▶ Новый код – это код, разработанный для нового приложения, который не включает большие порции ранее написанного кода
- ▶ Модифицируемый код – это код, разработанный для предыдущих приложений, который будет пригоден для использования в новом приложении после внесения умеренного объема изменений
- ▶ Повторно используемый код – это код, разработанный для предыдущих приложений, который будет пригодным для новых приложений без внесения каких-либо изменений

Первый этап при оценке систем, в которых возможно повторное использование кода, заключается в отделении нового кода от модифицируемого и повторно используемого кода. Это необходимо, так как интеграция модифицируемого и повторно используемого кода требует дополнительного объема трудозатрат



Рекомендации по выделению повторно используемого кода

- ▶ В качестве оцениваемой единицы выбирается программный модуль (примерный размер около 100 LOC)
- ▶ Если единица не была изменена она принимается за повторно используемый код
- ▶ Если единица была изменена (неважно идет ли речь об исполняемом коде, либо о комментариях) она считается модифицируемой
- ▶ Если объем изменений затрагивает более 50% кода единицы она принимается за новую
- ▶ В модифицируемом коде различают изменения, направленные на устранение дефектов и изменения, направленные на расширение функциональности



Пример, показывающий распределение новых модифицируемых и повторно используемых строк кода

Элемент	LOC для нового кода	LOC для модифицируемого кода	LOC для повторно используемого кода	Итого LOC
Компонент 1	1233	0	0	1233
Компонент 2	0	988	0	988
Компонент 3	0	0	781	781
Компонент 4	560	245	0	805
Компонент 5	345	549	420	1314
ИТОГО	2138	1782	1201	5121



Типичные множители повторного использования кода

Степень простоты использования	Повторно используемый код	Модифицируемый код
Простой	10%	25%
Средний	30%	60%
Сложный	40%	75%

Применение множителей повторного использования кода

Элемент	LOC для нового кода	LOC для модифицируемого кода	LOC для повторно используемого кода	Итого LOC
Итого	2138	1782	1201	5121
Множитель	100%	60%	30%	
Результат	2138	1069	360	3567



СОСМО II

Проект СОСМО II стартовал в 1995 году в центре по разработке ПО USC при финансовой и технической поддержке большого количества промышленных предприятий, таких как AT&T, BELL Labs, Hewlett-Packard, Rational, Texas Instruments, Lockheed Martin, Motorola, Xerox и др.

Проект имел триединую задачу:

- ▶ разработать модель для оценки стоимости и сроков создания ПО для того жизненного цикла, который будет применяться в конце 20 – начале 21 века
- ▶ разработать базу данных стоимости программных проектов и осуществить инструментальную поддержку методов усовершенствования модели стоимости
- ▶ создать количественную аналитическую схему для оценки технологий создания ПО и их экономического эффекта



Три различные модели оценки стоимости в СОСОМО II

- ▶ Модель композиции приложения – это модель, которая подходит для проектов, созданных с помощью современных инструментальных средств. Единицей измерения служит объектная точка
- ▶ Модель ранней разработки архитектуры. Эта модель применяется для получения приблизительных оценок проектных затрат периода выполнения проекта перед тем как будет определена архитектура в целом. В этом случае используется небольшой набор новых драйверов затрат и новых уравнений оценки. В качестве единиц измерения используются функциональные точки либо KSLOC
- ▶ Постархитектурная модель – наиболее детализированная модель СОСОМО II, которая используется после разработки архитектуры проекта. В состав этой модели включены новые драйверы затрат, новые правила подсчета строк кода, а также новые уравнения



Характеристики моделей оценки стоимости СОСМО II

Модель композиции приложения	Модель ранней разработки архитектуры	Постархитектурная модель
Грубые входные данные	Ясно понимаемые особенности проекта	Детальное описание проекта
Оценки низкой точности	Оценки умеренной точности	Высокоточные оценки
Приблизительные требования	Ясно понимаемые требования	Стабилизировавшиеся основные требования
Концепция архитектуры	Ясно понимаемая архитектура	Стабильная базовая архитектура



Модель композиции приложения

Данная модель используется на ранней стадии конструирования ПО, когда:

- ▶ рассматривается макетирование пользовательских интерфейсов
- ▶ оценивается производительность
- ▶ определяется степень зрелости технологии

Модель ориентирована на применение объектных точек. Объектная точка — средство косвенного измерения ПО. Подсчет количества объектных точек производится с учетом количества экранов (как элементов пользовательского интерфейса), отчетов и компонентов, требуемых для построения приложения



Правила подсчета объектных точек

- ▶ количество изображений на дисплее (экранных форм). Простые изображения принимаются за 1 объектную точку, изображения умеренной сложности принимаются за 2 точки, очень сложные изображения принято считать за 3 точки
- ▶ количество представленных отчетов. Для простых отчетов назначаются 2 объектные точки, умеренно сложным отчетам назначаются 5 точек. Написание сложных отчетов оценивается в 8 точек
- ▶ количество модулей, которые написаны на языках третьего поколения и разработаны в дополнение к коду, написанному на языке программирования четвертого поколения. Каждый модуль на языке третьего поколения считается за 10 объектных точек



Модель композиции приложения

NOP = (Объектные точки) x [(100 - %RUSE) / 100] – новые объектные точки

ТРУДОЗАТРАТЫ = NOP / PROD [чел.-мес.]

PROD - оценка скорости разработки

Модель композиции приложения используется на этапе создания прототипов и анализа осуществимости

Опытность/ возможности разработчика	Зрелость/ возможности среды разработки	PROD
Очень низкая	Очень низкая	4
Низкая	Низкая	7
Номинальная	Номинальная	13
Высокая	Высокая	25
Очень высокая	Очень высокая	50



Пример использования модели композиции приложения

Найти трудозатраты и календарное время работы над следующим проектом: приложение формирует 2 формы средней сложности (запросы к базе данных), 3 отчета высокой сложности и имеет 2 программных модуля на языке 3 уровня. Процент повторного использования кода программы – 10%. Над проектом работает программист средней квалификации, однако имеющий опыт работы в данной предметной области

3. Определим длительность выполнения проекта на уровне композиции приложения:

$$\text{Время} = 3 * (\text{Трудозатраты})^{(0.33 + 0.2 * (p-1.01))} = 3 * 3,32^{(0.33 + 0.2 * (p-1.01))} = 3 * 1,482 = 4,45 \text{ (месяца)},$$

где $p = 1$

1. Найдем количество объектных точек и их суммарную балльную оценку

№	Наименование объекта	Уровень сложности (коэф-нт)	Количество	Число точек
1	Форма	Средний (2)	2	4
2	Отчет	Высокий (8)	3	24
3	Модуль		2	20
всего			7	48

2. Определим трудозатраты на уровне композиции приложения, приняв среднюю производительность программиста 13 точек в месяц:

$$\text{Трудозатраты} = (\text{NOP} * (100 - \text{RUSE}) / 100) / \text{PROD} = (48 * (100 - 10) / 100) / 13 = 3,32 \text{ (чел./мес.)}$$



Модель ранней разработки архитектуры

Трудозатраты = 2,45*EArch*(Размер)^р,

где: Трудозатраты (работа) — число человеко-месяцев

EArch = PERS * RCPX * RUSE * PDIF * PREX * FCIL * SCED

Размер — KSLOC (предпочтительно для подсчета KSLOC предварительно подсчитать количество функциональных точек)

р — показатель степени

Время = 3,0 * (Трудозатраты) $(0.33 + 0.2 * (p-1.01))$

Примечание: Множитель EArch является произведением семи показателей, характеризующих проект и процесс создания ПО, а именно: надежность и уровень сложности разрабатываемой системы (RCPX), повторное использование компонентов (RUSE), сложность платформы разработки (PDIF), возможности персонала (PERS), опыт персонала (PREX), график работ (SCED) и средства поддержки (FCIL). Каждый множитель может быть оценен эксперто, либо его можно вычислить путем комбинирования значений более детализированных показателей, которые используются на постархитектурном уровне (см. след. слайд)



Модель ранней разработки архитектуры

Множитель трудоемкости	Идентификатор	Составные драйверы затрат
Надежность и сложность продукта	RCPX	RELY-DATA-CPLX-DOCU
Повторное использование компонентов	RUSE	RUSE
Сложность платформы	PDIF	TIME-STOR-PVOL
Опытность персонала	PERS	AEXP-PEXP-LTEX
Способности персонала	PREX	ACAP-PCAP-PCON
Возможности среды	FCIL	TOOL-SITE
Сроки	SCED	SCED

Примечание: высокий опыт персонала и интенсивное повторное использование компонентов ведут к снижению затрат



Значения множителей трудоемкости, в зависимости от их уровня

	Оценка уровня множителя трудоемкости					
	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверхвысокий
PERS	1.62	1.26	1.00	0.83	0.63	0.5
RCPX	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	n/a	0.95	1.00	1.07	1.15	1.24
PDIF	n/a	0.87	1.00	1.29	1.81	2.61
PREX	1.33	1.22	1.00	0.87	0.74	0.62
FCIL	1.30	1.10	1.00	0.87	0.73	0.62
SCED	1.43	1.14	1.00	1.00	1.00	n/a



Модель этапа постархитектуры

Трудозатраты = 2,45*ЕApp*(Размер)^p,

где:

Трудозатраты (работа) — число человеко-месяцев;

ЕApp — результат применения семнадцати уточняющих факторов постархитектурных этапов разработки

Время = 3,0 * (Трудозатраты) $(0.33 + 0.2 * (p-1.01))$



Усовершенствованная постархитектурная модель СОСОМО II

Идентификатор	Уточняющий фактор работ	Изменение в СОСОМО II
RELY	Требуемая надежность	Без изменений относительно СОСОМО
DATA	Размер базы данных	Без изменений относительно СОСОМО
CPLX	Сложность продукта	Без изменений относительно СОСОМО
RUSE	Требуемый уровень повторного использования	
DOCU	Документация	Добавлен. Определяет насколько документация соответствует требованиям жизненного цикла
TIME	Ограничение времени выполнения	Без изменений относительно СОСОМО
STOR	Ограничение объема основной памяти	Без изменений относительно СОСОМО



Усовершенствованная постархитектурная модель СОСОМО II

Идентификатор	Уточняющий фактор работ	Изменение в СОСОМО II
PVOL	Изменчивость платформы	Фактор изменчивости платформы
ACAP	Способности аналитика	Без изменений относительно СОСОМО
AEXP	Знание приложений	Без изменений относительно СОСОМО
PCAP	Способности программиста	Без изменений относительно СОСОМО
PCON	Преемственность персонала	Новый параметр
LTEX	Знание языка программирования и инструментария	Изменен с целью охвата знаний инструментария и языка
SITE	Распределенная разработка. Взаимодействие между командами разработчиков	Новые параметры, определяющие степень взаимной удаленности команд разработчиков и степень автоматизации их деятельности
TOOL	Использование программных инструментов	Без изменений относительно СОСОМО
SCED	Требуемые сроки разработки	Без изменений относительно СОСОМО

Правила вычисления показателя степени в модели СОСОМО II

- ▶ Значение показателя степени изменяется от 1.1 до 1.24
- ▶ Значение показателя степени зависит от того, насколько новаторским является данный проект, от гибкости процесса разработки ПО, от применяемых процессов управления рисками, сплоченности команды программистов и уровня управления организацией-разработчиком
- ▶ Значение показателя степени рассчитывается с учетом пяти показателей по восьмибалльной шкале от низшего (7 баллов) до наивысшего (0 баллов) уровня
- ▶ Значения всех пяти показателей суммируются, сумма делится на 100, результат прибавляется к числу 1.01



Факторы, влияющие на показатель степени в модели СОСМО II

- ▶ Наличие прецедентов у приложения: уровень опыта организации-разработчика в данной области
- ▶ Гибкость процесса: степень строгости контракта, порядок его выполнения, присущая контракту свобода внесения изменений, виды деятельности в течение жизненного цикла и взаимодействие между заинтересованными сторонами
- ▶ Разрешение рисков, присущих архитектуре: степень технической осуществимости, продемонстрированной до перехода к полномасштабному производству
- ▶ Сплоченность команды: степень сотрудничества и того, насколько все заинтересованные стороны (покупатели, разработчики, пользователи, ответственные за сопровождение и др.) разделяют общую концепцию
- ▶ Зрелость процесса: уровень зрелости организации-разработчика, определяемая в соответствии с моделью СММ



Новизна проекта (PREC)

Отражает предыдущий опыт организации в реализации проектов данного типа. Очень низкий уровень этого показателя означает отсутствие опыта, наивысший уровень указывает на компетентность организации-разработчика в данной области ПО

Характеристика	Рекомендуемое значение
Полное отсутствие прецедентов, полностью непредсказуемый проект	6,2
Почти полное отсутствие прецедентов, в значительной мере непредсказуемый проект	4,96
Наличие некоторого количества прецедентов	3,72
Общее знакомство с проектом	2,48
Значительное знакомство с проектом	1,24
Полное знакомство с проектом	0



Гибкость процесса разработки (FLEX)

Отображает возможность изменения процесса разработки ПО. Очень низкий уровень этого показателя означает, что процесс определен заказчиком заранее, наивысший — заказчик определил лишь общие задачи без указания конкретной технологии процесса разработки ПО

Характеристика	Рекомендуемое значение
Точный, строгий процесс разработки	5,07
Случайные послабления в процессе	4,05
Некоторые послабления в процессе	3,04
Большой частью согласованный процесс	2,03
Некоторое согласование процесса	1,01
Заказчик определил только общие цели	0



Разрешение рисков в архитектуре системы (RESL)

Отображает степень детализации анализа рисков, основанного на анализе архитектуры системы. Очень низкий уровень данного показателя соответствует поверхностному анализу рисков, наивысший уровень означает, что был проведен тщательный и полный анализ всевозможных рисков

Характеристика	Рекомендуемое значение
Малое (20 %)	7
Некоторое (40 %)	5,65
Частое (60 %)	4,24
В целом (75 %)	2,83
Почти полное (90 %)	1,41
Полное (100%)	0



Сплоченность команды (TEAM)

Отображает степень сплоченности команды и их способность работать совместно. Очень низкий уровень этого показателя означает, что взаимоотношения в команде сложные, а наивысший — что команда сплоченная и эффективная в работе, не имеет проблем во взаимоотношениях

Характеристика	Рекомендуемое значение
Сильно затрудненное взаимодействие	5,48
Несколько затрудненное взаимодействие	4,38
Некоторая согласованность	3,29
Повышенная согласованность	2,19
Высокая согласованность	1,1
Взаимодействие как в едином целом	0



Уровень зрелости процесса разработки (РМАТ)

Отображает уровень развития процесса создания ПО в организации-разработчике

Характеристика	Рекомендуемое значение
Уровень 1 СММ	7
Уровень 1+ СММ	6,24
Уровень 2 СММ	4,68
Уровень 3 СММ	1,12
Уровень 7 СММ	1,56
Уровень 5 СММ	0



Пример использования модели СОСМО II

Предположим, что организация-разработчик выполняет программный проект в той области, в которой у нее мало опыта разработок. Заказчик ПО не определил строгий технологический процесс, который будет использоваться при создании программного продукта, но некоторые согласования были проведены. Заказчик не выделил в плане работ времени на анализ возможных рисков в архитектуре системы. Для создания программной системы необходимо сформировать новую команду специалистов. Организация-разработчик недавно привела в действие программу усовершенствования технологического процесса разработки ПО и может котироваться как организация второго уровня в соответствии с моделью оценки уровня зрелости.

Предварительный размер проекта оценен в 128 KSLOC.

Предполагается очень высокая надежность системы и сложность системных модулей, высокие ограничения объема памяти, низкий уровень использования программных инструментов и ускоренный график работы. Остальные драйверы затрат номинальные



Показатели проекта

Показатель проекта	Характеристика показателя	Значение
Новизна проекта	Это новый проект для организации, но предметная область является знакомой; данный показатель имеет низкий уровень	3,72
Гибкость процесса разработки	Некоторое согласование процесса разработки с заказчиком дает высокое значение показателю	1,01
Анализ архитектуры системы и рисков	Анализ не был проведен — уровень данного показателя очень низкий	7
Сплоченность команды	Команда разработчиков новая, информация о ней отсутствует - уровень этого показателя оценивается как обычный	3,29
Уровень развития процесса разработки	Организация имеет второй уровень зрелости процессов разработки	4,68
Итого		19,7

Значение показателя степени $r = 19,7/100 + 1,01 = 1,207$



Выполнение расчетов

Трудозатраты = $2,45 * EApp * (\text{Размер})^p$

$$Eapp = RELY * CPLX * STOR * TOOL * SCED = 1,25 * 1,3 * 1,21 * 1,12 * 1,29 = 2,841$$

$$\text{Трудозатраты} = 2,45 * 2,841 * 128^{1,207} = 2432 \text{ чел.-мес.}$$

$$\text{Время} = 3 * (\text{Трудозатраты})^{(0.33 + 0.2 * (p-1.01))} =$$

$$= 3 * 2432^{(0.33 + 0.2 * (1,207 - 1,01))} = 53,44 \text{ мес.}$$



Достоинства модели СОСМО II

- ▶ возможен учет достаточно полной номенклатуры факторов, влияющих на экономические характеристики производства сложных программных продуктов
- ▶ метод является достаточно универсальным и может поддерживать различные размеры, режимы и уровни качества продуктов
- ▶ фактические данные подбираются в соответствии с реальными проектами и факторами корректировки, которые могут соответствовать конкретному проекту и организации
- ▶ прогнозы производственных процессов являются варьируемыми и повторяемыми
- ▶ метод позволяет добавлять уникальные факторы для корректировки экономических характеристик, связанные со специфическим проектом и организацией
- ▶ возможна высокая степень достоверности калибровки с опорой на предыдущий опыт коллектива специалистов
- ▶ результаты прогнозирования сопровождаются обязательной документацией
- ▶ модель относительно проста в освоении и применении



Недостатки модели СОСМО II

- ▶ все результаты зависят от размера программного продукта: точность оценки размера, оказывает определяющее влияние на точность прогноза трудозатрат, длительности разработки и численности специалистов
- ▶ игнорируются требования к характеристикам качества программного продукта
- ▶ не учитывается зависимость между интегральными затратами и количеством времени, затрачиваемым на каждом этапе проекта
- ▶ игнорируется изменяемость требований к программному продукту в процессе производства
- ▶ недостаточно учитывается внешняя среда производства и применения программного продукта
- ▶ игнорируются многие особенности, связанные с аппаратным обеспечением проекта

Модель СОСМО II изначально представляет трудозатраты по сумме всех этапов производства (от этапа планирования концепции до этапа поставки программного продукта). При этом не учитываются проблемы сопровождения, переноса и повторного использования компонентов, которые трудно описывать в рамках одной и той же модели



Последовательное уточнение технико-экономических параметров программного проекта

1. Экспертные оценки экономических характеристик по прототипам – достоверность оценивания размера продукта 40 – 50%

2. Этап предварительного проектирования

- ▶ оценивание экономических характеристик в базовой модели СОСОМО – достоверность оценивания размера продукта 20 – 30%
- ▶ оценивание экономических характеристик в детализированной модели СОСОМО – достоверность оценивания размера продукта 10 – 20%

3. Этап детального проектирования

- ▶ оценивание экономических характеристик в упрощенной предварительной модели СОСОМО достоверность оценивания размера продукта 5 – 10% (учитывается 7 параметров)
- ▶ оценивание экономических характеристик в детальной модели СОСОМО II – достоверность оценивания размера продукта 3 – 5% (учитывается 17 параметров)





Спасибо за внимание!

Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана
Каф. ИУ-7

baryshnikovam@mail.ru

Лекция 8

Оценка размера программного продукта на основе метода функциональных точек

Хорошей считается оценка, которая обеспечивает достаточно ясное представление реального состояния проекта и позволяет руководителю проекта принимать хорошие решения относительно того, как управлять проектом для достижения целей

Стив Макконнелл



Функционально-ориентированные метрики измерения программного продукта

- ▶ Функционально-ориентированные метрики косвенно измеряют программный продукт и процесс его разработки
- ▶ При этом рассматривается не размер, а функциональность или полезность продукта
- ▶ В качестве количественной характеристики применяется понятие количества функциональных точек FP (function points)

Для получения функционально-ориентированных метрик (FP-метрик) используются функциональные и концептуальные модели будущей системы (например, модель IDEF0). При этом рассматриваются только наиболее значимые процессы, соответствующие основным функциям разрабатываемого программного продукта (например, перечисленным в техническом задании)



Метод функциональных точек

Функциональная точка — это единица измерения функциональности программного обеспечения. Функциональность программы связана с обработкой информации по запросу пользователя и не зависит от применяемых технических решений.

Пользователи — это отправители и целевые получатели данных, ими могут быть как реальные люди, так и смежные интегрированные информационные системы

Метод функциональных точек позволяет:

- ▶ оценивать категории пользовательских бизнес-функций
- ▶ разрешить проблему, связанную с трудностью получения LOC – оценок на ранних стадиях жизненного цикла
- ▶ определять количество и сложность входных и выходных данных, их структуру, а также внешние интерфейсы, связанные с программной системой



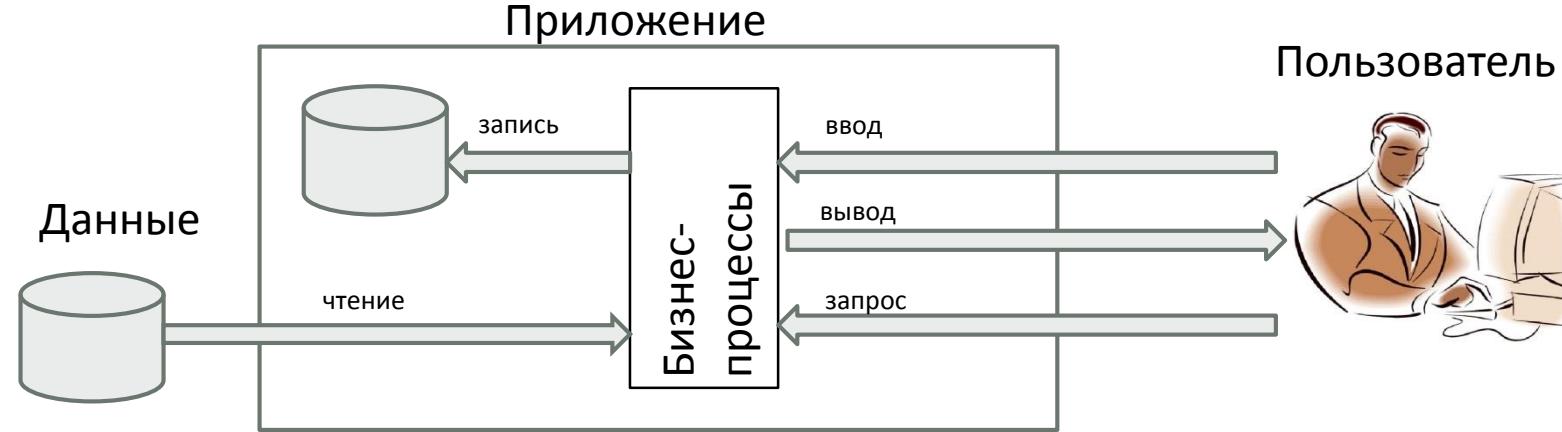
Метод функциональных точек: историческая справка

- ▶ Функциональные точки (function points) были впервые предложены в качестве метрики измерения размера программного продукта сотрудником компании IBM Алланом Альбрехтом (Allan Albrecht) в 1979 г.
- ▶ Применение функциональных точек основано на изучении требований, поэтому оценка необходимых трудозатрат может быть выполнена на самых ранних стадиях работы над проектом и далее будет уточняться по ходу жизненного цикла
- ▶ Явная связь между требованиями к создаваемой системе и получаемой оценкой позволяет заказчику понять, за что именно он платит, и во что выльется изменение первоначального задания
- ▶ Постепенно метод функциональных точек превратился в индустриальный стандарт, и в 1986 г. для его поддержки и развития была создана некоммерческая организация IFPUG (International Function Point User Group)



Методика оценки трудоемкости разработки ПО на основе функциональных точек

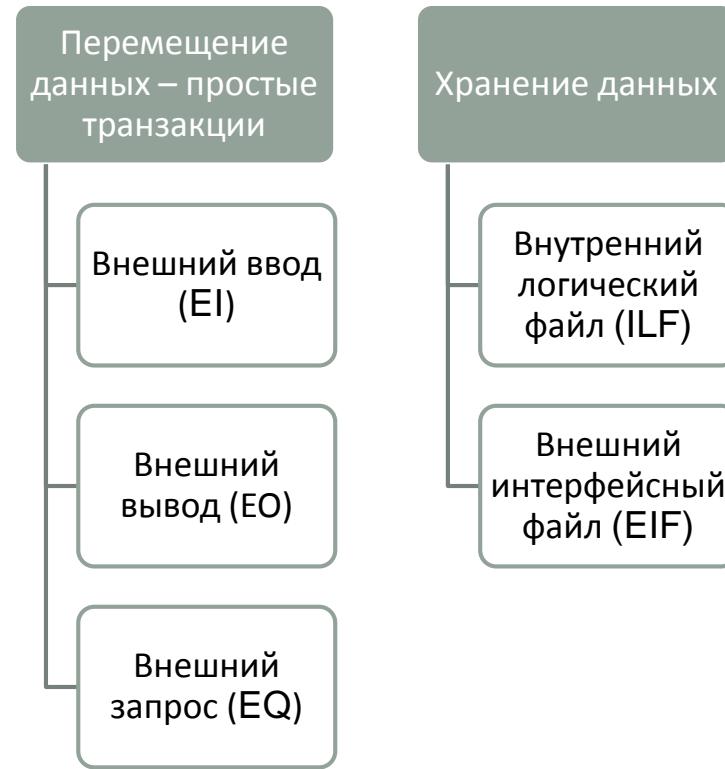
Определение числа функциональных точек является методом количественной оценки ПО, применяемым для измерения функциональных характеристик процессов его разработки и сопровождения независимо от технологии, использованной для его реализации



Трудоемкость вычисляется на основе функциональности разрабатываемой системы, которая, в свою очередь, определяется путем выявления **функциональных типов** — логических групп взаимосвязанных данных, используемых и поддерживаемых приложением, а также **элементарных процессов**, связанных с вводом и выводом информации



Функциональные типы данных и типы элементарных процессов, используемые в методе функциональных точек



Транзакция — это элементарный процесс, различаемый пользователем и перемещающий данные между внешней средой и программным приложением. В своей работе транзакции используют внутренние и внешние файлы

Внешний ввод (EI, транзакция, получающая данные от пользователя) — элементарный процесс, перемещающий данные из внешней среды в приложение. Данные могут поступать с экрана ввода или из другого приложения. Данные могут содержать как управляющую, так и деловую информацию. Обрабатываемые данные могут соответствовать одному или нескольким внутренним логическим файлам

Внешний вывод (EO, транзакция передающая данные пользователю) — элементарный процесс, перемещающий данные, вычисленные в приложении, во внешнюю среду и связанный с созданием и/или обработкой выходной информации приложения — выходного отчета, документа, экранной формы

Внешний запрос (EQ, интерактивный диалог с пользователем, требующий от него каких-либо действий) — элементарный процесс, состоящий из комбинации «запрос/ответ», не связанный с вычислением производных данных или обновлением внутренних логических файлов (базы данных)

Внутренний логический файл (ILF, информация, которая используется во внутренних взаимодействиях системы) — выделяемые пользователем логически связанные группы данных или блоки управляющей информации, которые поддерживаются внутри продукта и обслуживается через внешние вводы

Внешний интерфейсный файл (EIF, файлы, участвующие во внешних взаимодействиях с другими системами) — выделяемые пользователем логически связанные группы данных или блоки управляющей информации, на которые ссылается продукт, но которые поддерживаются вне продукта

Примеры

Информационная характеристика	Элементы данных
Внешние вводы	Поля ввода данных, сообщения об ошибках, вычисляемые значения, кнопки
Внешние выводы	Поля данных в отчетах, вычисляемые значения, сообщения об ошибках, заголовки столбцов, которые читаются из внутреннего файла
Внешние запросы	Вводимые элементы: поле, используемое для поиска, щелчок мыши. Выводимые элементы — отображаемые на экране поля



Информационные характеристики, используемые в методе функциональных точек

- ▶ *Количество внешних вводов.* Подсчитываются все вводы пользователя, по которым поступают разные прикладные данные, при этом вводы должны быть отделены от запросов, которые подсчитываются отдельно
- ▶ *Количество внешних выводов.* Подсчитываются все выводы, по которым к пользователю поступают результаты, вычисленные программным приложением. В этом контексте выводы означают отчеты, экраны, распечатки, сообщения об ошибках. Индивидуальные единицы данных внутри отчета отдельно не подсчитываются
- ▶ *Количество внешних запросов.* Под запросом понимается диалоговый ввод, который приводит к немедленному программному ответу в форме диалогового вывода. При этом диалоговый ввод в приложении не сохраняется, а диалоговый вывод не требует выполнения вычислений. Подсчитываются все запросы — каждый учитывается отдельно
- ▶ *Количество внутренних логических файлов.* Подсчитываются все логические файлы (то есть логические группы данных, которые могут быть частью базы данных или отдельным файлом)
- ▶ *Количество внешних интерфейсных файлов.* Подсчитываются все логические файлы из других приложений, на которые ссылается данное приложение

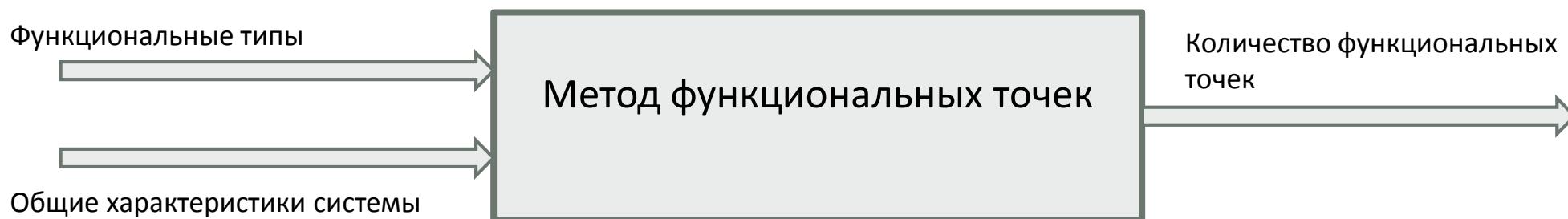


Как связаны информационные характеристики?

- ▶ ILF и EIF являются логически связанными группами данных (файлами, таблицами баз данных и т.п.), определяемыми пользователем и необходимыми для работы рассматриваемого программного средства. Оба эти типа файлов могут использоваться при генерации внешних выводов (EO), а также при обслуживании внешних запросов (EQ). Разница между ними состоит лишь в том, что ILF поддерживаются самим рассматриваемым ПС путем использования внешних вводов (EI), а EIF поддерживаются каким-либо другим приложением
 - ▶ Функциональность логических файлов (ILF и EIF) оценивается путем подсчета количества типов элементов записей (RET) и количества типов элементов данных (DET), входящих в соответствующие логические группы данных. При этом под количеством RET обычно понимается количество различных логических подгрупп данных, выделяемых в файле с точки зрения пользователя, или количество различных используемых форматов записей, а под количеством DET - количество различных элементарных полей в этих записях
 - ▶ Основным источником информации для оценки количества RET и DET в логическом файле могут служить, например, словарь данных (Data Dictionary), составленный в ходе моделирования информационной системы с использованием диаграмм потоков данных, логическая структура баз данных, полученная при построении моделей «сущность-связь», логическая структура файлов или баз данных, поддерживаемых другими приложениями и описанная в их документации (для EIF), и т.п.
-

Порядок расчета трудоемкости разработки ПО

- ▶ определение количества функциональных типов приложения
- ▶ определение количества связанных с каждым функциональным типом элементарных данных (DET), элементарных записей (RET) и файлов типа ссылок (FTR)
- ▶ определение сложности (в зависимости от количества DET, RET и FTR)
- ▶ подсчет количества функциональных точек приложения
- ▶ подсчет количества функциональных точек с учетом общих характеристик системы
- ▶ оценка трудоемкости разработки (с использованием различных статистических данных)



Определение количества и сложности транзакционных функциональных типов

- ▶ Количество транзакционных функциональных типов (входных элементов приложения, выходных элементов приложения и внешних запросов) определяется на основе выявления входных и выходных документов, экраных форм, отчетов, а также по диаграммам классов
- ▶ Далее для каждого выявленного функционального типа (EI, EO или EQ) определяется его сложность (низкая, средняя или высокая), которая зависит от количества связанных с этим функциональным типом DET, RET и FTR

Еще раз про DET, RET, FTR

- ▶ DET – это уникальное распознаваемое пользователем, нерекурсивное (неповторяющееся) поле данных
 - ▶ RET - идентифицируемая пользователем логическая группа данных внутри ILF или EIF
 - ▶ FTR – это тип файла, на который ссылается транзакция. FTR позволяет подсчитать количество различных файлов (информационных объектов) типа ILF и/или EIF модифицируемых или считываемых в транзакции
-

Ранг и оценка сложности внутренних логических файлов

Типы элементов-записей (RET)	Элементы данных (DET)		
	1-19	20-50	>50
1	Низкий (7)	Низкий (7)	Средний (10)
2-5	Низкий (7)	Средний (10)	Высокий (15)
>5	Средний (10)	Высокий (15)	Высокий (15)

Ранг и оценка сложности внешних интерфейсных файлов

Типы элементов-записей (RET)	Элементы данных (DET)		
	1-19	20-50	>50
1	Низкий (5)	Низкий (5)	Средний (7)
2-5	Низкий (5)	Средний (7)	Высокий (10)
>5	Средний (7)	Высокий (10)	Высокий (10)



Правила расчета DET для внешних вводов

В качестве DET (data element types) для внешних вводов (EI) учитываются:

- ▶ каждое нерекурсивное поле, принадлежащее внутреннему логическому файлу (ILF) (или поддерживаемое им) и обрабатываемое во вводе
- ▶ каждое поле, которое пользователь хотя и не вызывает, но оно через процесс ввода поддерживается во внутреннем логическом файле (ILF)
- ▶ логическое поле, которое физически представляет собой множество полей, но воспринимается пользователем как единый блок информации
- ▶ группа полей, которые появляются во внутреннем логическом файле (ILF) более одного раза, но в связи с особенностями алгоритма их использования воспринимаются как один DET
- ▶ группа полей, которые фиксируют ошибки в процессе обработки или подтверждают, что обработка закончилась успешно
- ▶ действие, которое может быть выполнено во вводе

Ранг и оценка сложности внешних вводов

Ссылки на файлы (FTR)	Элементы данных (DET)		
	1-4	5-15	>15
0-1	Низкий (3)	Низкий (3)	Средний (4)
2	Низкий (3)	Средний (4)	Высокий (6)
>2	Средний (4)	Высокий (6)	Высокий (6)



Правила расчета DET для внешних выводов

В качестве DET для внешних выводов (EO) учитываются:

- ▶ каждое распознаваемое пользователем нерекурсивное поле, участвующее в процессе вывода
- ▶ поле, которое физически отображается в виде нескольких полей его составляющих, но используется как единый информационный элемент
- ▶ каждый тип метки и каждое значение числового эквивалента при графическом выводе
- ▶ текстовая информация, которая может содержать одно слово, предложение или фразу

Примечание: переменные, определяющие номера страниц или генерируемые системой логотипы не являются элементами данных

Ранг и оценка сложности внешних выводов

Ссылки на файлы (FTR)	Элементы данных (DET)		
	1-4	5-19	>19
0-1	Низкий (4)	Низкий (4)	Средний (5)
2-3	Низкий (4)	Средний (5)	Высокий (7)
>3	Средний (5)	Высокий (7)	Высокий (7)



Правила расчета DET для внешних запросов

В качестве DET для внешнего запроса (EQ) **по входу** учитываются:

- ▶ каждое распознаваемое пользователем нерекурсивное поле, появляющееся во вводной части запроса
- ▶ каждое поле, которое определяет критерий выбора данных
- ▶ группа полей, в которых выдаются сообщения о возникающих ошибках в процессе ввода информации или подтверждающих успешное завершение процесса ввода
- ▶ группа полей, которые позволяют выполнять запросы

В качестве DET для внешнего запроса (EQ) **по выходу** учитываются:

- ▶ каждое распознаваемое пользователем нерекурсивное поле, которое появляется в выводной части запроса
- ▶ логическое поле, которое физически отображается как группа полей, однако воспринимается пользователем как единое поле
- ▶ группа полей, которые в соответствии с методикой обработки могут повторяться во внутреннем логическом файле (ILF)

Примечание: колонтитулы или генерируемые системой иконки не учитываются как DET



Ранг и оценка сложности внешних запросов

Ссылки на файлы (FTR)	Элементы данных		
	1-4	5-19	>19
0-1	Низкий (3)	Низкий (3)	Средний (4)
2-3	Низкий (3)	Средний (4)	Высокий (6)
>3	Средний (4)	Высокий (6)	Высокий (6)



Учет элементов данных из графического интерфейса пользователя

Элемент данных	Правило учета
Группа радиокнопок	Так как в группе пользователь выбирает только одну радиокнопку, все радиокнопки группы считаются одним элементом данных
Группа флагков (переключателей)	Так как в группе пользователь может выбрать несколько флагков, каждый флагок считают элементом данных
Командные кнопки	Командная кнопка может определять действие добавления, изменения или запроса. Кнопка ОК может вызывать транзакции различных типов. Каждая кнопка считается отдельным элементом данных
Списки	Список может быть внешним запросом, но результат запроса может быть элементом данных внешнего ввода



Алгоритм применения метода функциональных точек

- ▶ Подсчитываются функции для каждой категории (вводы, выводы, запросы, структуры данных, интерфейсы)
- ▶ Устанавливаются требования для каждой категории
- ▶ Определяется сложность каждой функции (высокая, средняя, низкая)
- ▶ Каждая функция умножается на соответствующий ей параметр, а затем суммируется с целью получения общего количества функциональных точек

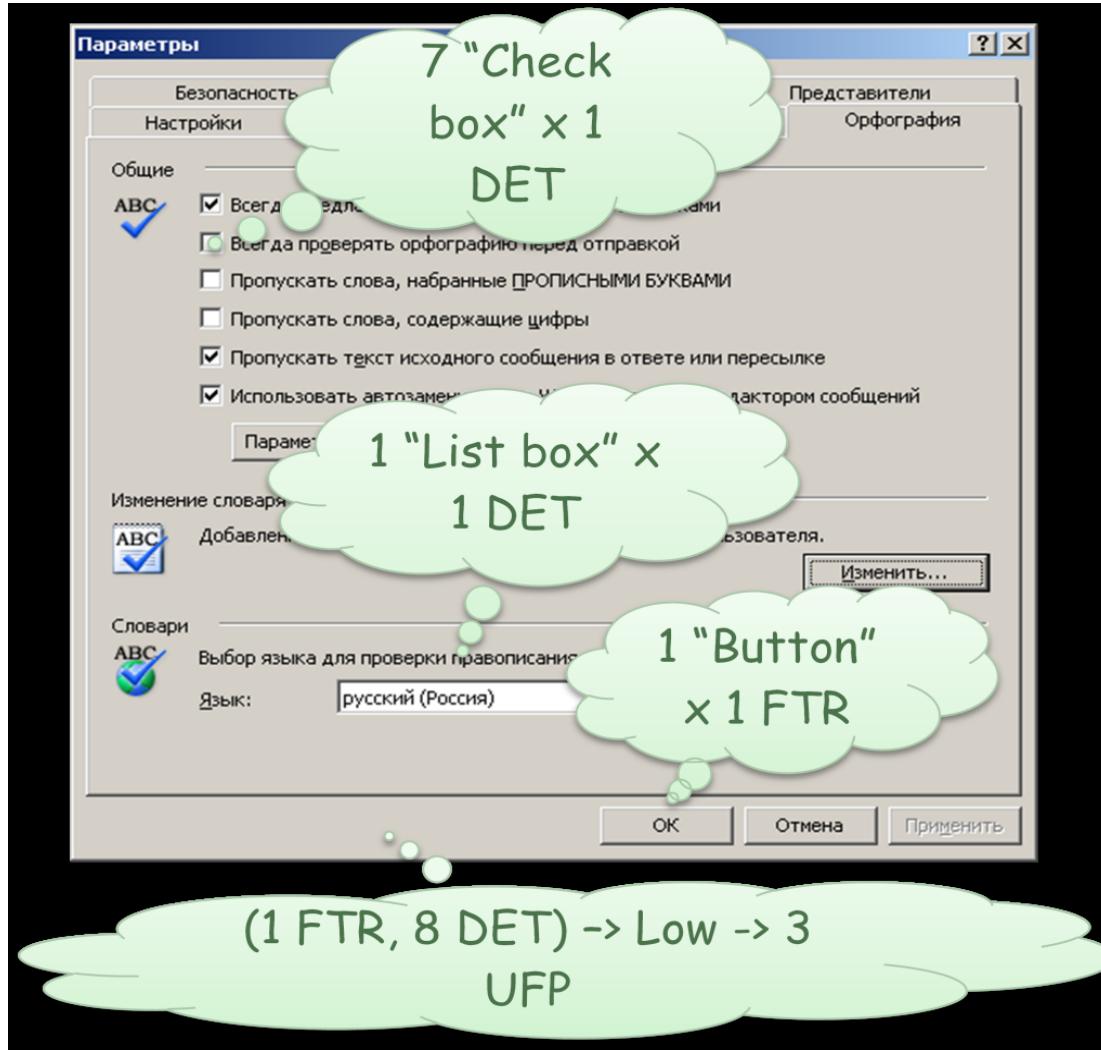


Исходные данные для расчета общего количества функциональных точек

Имя характеристики	Ранг, сложность, количество			
	Низкий	Средний	Высокий	Итого
Внешние вводы	$_x3 = \underline{\hspace{2cm}}$	$_x4 = \underline{\hspace{2cm}}$	$_x6 = \underline{\hspace{2cm}}$	$= \underline{\hspace{2cm}}$
Внешние выводы	$_x4 = \underline{\hspace{2cm}}$	$_x5 = \underline{\hspace{2cm}}$	$_x7 = \underline{\hspace{2cm}}$	$= \underline{\hspace{2cm}}$
Внешние запросы	$_x3 = \underline{\hspace{2cm}}$	$_x4 = \underline{\hspace{2cm}}$	$_x6 = \underline{\hspace{2cm}}$	$= \underline{\hspace{2cm}}$
Внутренние логические файлы	$_x7 = \underline{\hspace{2cm}}$	$_x1 = \underline{\hspace{2cm}}$	$_x15 = \underline{\hspace{2cm}}$	$= \underline{\hspace{2cm}}$
Внешние интерфейсные файлы	$_x5 = \underline{\hspace{2cm}}$	$_x7 = \underline{\hspace{2cm}}$	$_x1 = \underline{\hspace{2cm}}$	$= \underline{\hspace{2cm}}$
Общее количество				$= \underline{\hspace{2cm}}$



Диалоговое окно, управляющее проверкой орфографии в MS Office Outlook



Источник: С. Архипенков. Лекции по управлению программными проектами, стр. 102

Примеры анализа данных

Проектируется GUI для обслуживания клиентов, который будет иметь поля: *Имя, Адрес, Город, Страна, Почтовый Индекс, Телефон, Email* (таким образом, имеется 7 полей или 7 DET)

Кроме того, предусматриваются 3 командные кнопки: *Добавить, Изменить, Удалить*. Следовательно каждый из внешних вводов Добавить, Изменить, Удалить будет состоять из 8 DET (7 полей плюс командная кнопка)

В данном приложении генерируются 3 типа сообщений: *сообщения об ошибке, сообщения подтверждения и сообщения уведомления*. Сообщения об ошибке (например, *Требуется пароль*) и сообщения подтверждения (например, *Вы действительно хотите удалить клиента?*) указывают, что произошла ошибка или что процесс может быть завершен. Эти сообщения не образуют самостоятельного процесса, они являются частью другого процесса, то есть считаются элементом данных соответствующей транзакции

Уведомление является независимым элементарным процессом. Например, при попытке получить из банкомата сумму денег, превышающую их количество на счете, генерируется сообщение *Не хватает средств для завершения транзакции*. Оно является результатом чтения информации из файла счета и формирования заключения. Сообщение уведомления рассматривается как внешний вывод



Скорректированное количество функциональных точек

$$FP = \text{Общее количество} * (0,65 + 0,01 * \sum Fi),$$

где Fi — 14 коэффициентов регулировки сложности

Каждый коэффициент может принимать следующие значения:

- 0 — нет влияния,
- 1 — случайное,
- 2 — небольшое,
- 3 — среднее,
- 4 — важное,
- 5 — основное



Определение системных параметров приложения

№	Системный параметр	Описание
1	Передача данных	Сколько средств связи требуется для передачи или обмена информацией с приложением или системой?
2	Распределенная обработка данных	Как обрабатываются распределенные данные и функции обработки?
3	Производительность	Нуждается ли пользователь в фиксации времени ответа или производительности?
4	Эксплуатационные ограничения	Насколько сильны эксплуатационные ограничения и каков объем специальных усилий на их преодоление?
5	Частота транзакций	Как часто выполняются транзакции (каждый день, каждую неделю, каждый месяц) ?
6	Оперативный ввод данных	Какой процент информации надо вводить в режиме онлайн?
7	Эффективность работы конечных пользователей	Приложение проектировалось для обеспечения эффективной работы конечного пользователя?
8	Оперативное обновление	Как много внутренних файлов обновляется в онлайновой транзакции?
9	Сложность обработки	Выполняет ли приложение интенсивную логическую или математическую обработку?
10	Повторная используемость	Приложение разрабатывалось для удовлетворения требований одного или многих пользователей?
11	Легкость инсталляции	Насколько трудны преобразование и инсталляция приложения?
12	Легкость эксплуатации	Насколько эффективны и/или автоматизированы процедуры запуска, резервирования и восстановления?
13	Количество возможных установок на различных платформах	Была ли спроектирована, разработана и поддержана возможность инсталляции приложения в разных местах для различных организаций?
14	Простота изменений (гибкость)	Была ли спроектирована, разработана и поддержана в приложении простота изменений?

Передачи данных

Значение	Описание
0	Полностью пакетная обработка на локальном ПК
1	Пакетная обработка, удаленный ввод данных или удаленная печать
2	Пакетная обработка, удаленный ввод данных и удаленная печать
3	Сбор данных в режиме «онлайн» или дистанционная обработка, связанная с пакетным процессом
4	Несколько внешних интерфейсов, один тип коммуникационного протокола
5	Несколько внешних интерфейсов, несколько типов коммуникационных протоколов



Распределенная обработка данных

Значение	Описание
0	Передача данных или процессов между компонентами системы отсутствует
1	Приложение готовит данные для обработки на ПК конечного пользователя
2	Данные готовятся для передачи, затем передаются и обрабатываются на другом компоненте системы (не на ПК конечного пользователя)
3	Распределенная обработка и передача данных в режиме «онлайн» только в одном направлении
4	Распределенная обработка и передача данных в режиме «онлайн» в обоих направлениях
5	Динамическое выполнение процессов в любом подходящем компоненте системы



Производительность

Значение	Описание
0	К системе не предъявляются специальных требований, касающихся производительности
1	Требования к производительности определены, но не требуется никаких специальных действий
2	Время реакции или пропускная способность являются критическими в пиковые периоды. Не требуется никаких специальных решений относительно использования ресурсов процессора. Обработка может быть завершена в течение следующего рабочего дня
3	Время реакции или пропускная способность являются критическими в обычное рабочее время. Не требуется никаких специальных решений относительно использования ресурсов процессора. Время обработки ограничено взаимодействующими системами
4	То же, кроме того, пользовательские требования к производительности достаточно серьезны, чтобы ее необходимо было анализировать на стадии проектирования
5	То же, кроме того, на стадиях проектирования, разработки и (или) реализации для удовлетворения пользовательских требований к производительности используются специальные средства анализа



Эксплуатационные ограничения

Значение	Описание
0	Какие-либо явные или неявные ограничения отсутствуют
1	Эксплуатационные ограничения присутствуют, но не требуют никаких специальных усилий
2	Должны учитываться некоторые ограничения, связанные с безопасностью или временем реакции
3	Должны учитываться конкретные требования к процессору со стороны конкретных компонентов приложения
4	Заданные эксплуатационные ограничения требуют специальных ограничений на выполнение приложения в центральном или выделенном процессоре
5	То же, кроме того, специальные ограничения затрагивают распределенные компоненты системы



Частота транзакций

Значение	Описание
0	Пиковых периодов не ожидается
1	Ожидается пиковые периоды (ежемесячные, ежеквартальные, ежегодные)
2	Ожидается еженедельные пиковые периоды
3	Ожидается ежедневные пиковые периоды
4	Высокая частота транзакций требует анализа производительности на стадии проектирования
5	То же, кроме того, на стадиях проектирования, разработки и (или) внедрения необходимо использовать специальные средства анализа производительности



Оперативный ввод данных

Значение	Описание
0	Все транзакции обрабатываются в пакетном режиме
1	От 1% до 7% транзакций требуют интерактивного ввода данных
2	От 8% до 15% транзакций требуют интерактивного ввода данных
3	От 16% до 23% транзакций требуют интерактивного ввода данных
4	От 24% до 30% транзакций требуют интерактивного ввода данных
5	Более 30% транзакций требуют интерактивного ввода данных



Эффективность работы конечных пользователей

Эффективность работы конечных пользователей определяется наличием следующих функциональных возможностей:

- ▶ Средства навигации (например, функциональные клавиши, динамически генерируемые меню)
- ▶ Меню
- ▶ Онлайновые подсказки и документация
- ▶ Автоматическое перемещение курсора
- ▶ Скроллинг
- ▶ Удаленная печать
- ▶ Предварительно назначенные функциональные клавиши
- ▶ Выбор данных на экране с помощью курсора
- ▶ Использование видеоэффектов, цветового выделения, подчеркивания и других индикаторов
- ▶ Всплывающие окна
- ▶ Минимизация количества экранов, необходимых для выполнения бизнес-функций
- ▶ Поддержка двух и более языков



Эффективность работы конечных пользователей

Значение	Описание
0	Ни одной из перечисленных функциональных возможностей
1	От одной до трех функциональных возможностей
2	От четырех до пяти функциональных возможностей
3	Шесть или более функциональных возможностей при отсутствии конкретных пользовательских требований к эффективности
4	То же, кроме того, пользовательские требования к эффективности требуют специальных проектных решений для учета эргономических факторов (например, минимизации нажатий клавиш, максимизации значений по умолчанию, использования шаблонов)
5	То же, кроме того, пользовательские требования к эффективности требуют применения специальных средств и процессов, демонстрирующих их выполнение



Оперативное обновление

Значение	Описание
0	Отсутствует
1	Онлайновое обновление от одного до трех управляющих файлов Объем обновлений незначителен, восстановление несложно
2	Онлайновое обновление четырех или более управляющих файлов Объем обновлений незначителен, восстановление несложно
3	Онлайновое обновление основных внутренних логических файлов
4	То же, плюс необходимость специальной защиты от потери данных
5	То же, кроме того, большой объем данных требует учета затрат на процесс восстановления. Требуются автоматизированные процедуры восстановления с минимальным вмешательством оператора



Сложность обработки

Сложность обработки характеризуется наличием у приложения следующих функциональных возможностей:

- ▶ повышенная реакция на внешние воздействия и (или) специальная защита от внешних воздействий
- ▶ экстенсивная логическая обработка
- ▶ экстенсивная математическая обработка
- ▶ обработка большого количества исключительных ситуаций
- ▶ поддержка разнородных типов входных/выходных данных

Значение	Описание
0	Ни одной из перечисленных функциональных возможностей
1	Любая одна из возможностей
2	Любые две возможности
3	Любые три возможности
4	Любые три возможности
5	Все пять возможностей



Повторное использование

Значение	Описание
0	Отсутствует
1	Повторное использование кода внутри одного приложения
2	Не более 10% приложений будут использоваться более чем одним пользователем
3	Более 10% приложений будут использоваться более чем одним пользователем
4	Приложение оформляется как продукт и (или) документируется для облегчения повторного использования. Настройка приложения выполняется пользователем на уровне исходного кода
5	То же, с возможностью параметрической настройки приложений



Легкость инсталляции

Значение	Описание
0	К установке не предъявляется никаких специальных требований
1	Для установки требуется специальная процедура
2	Заданы пользовательские требования к конвертированию (переносу существующих данных и приложений в новую систему) и установке, должны быть обеспечены и проверены соответствующие руководства. Конвертированию не придается важное значение
3	То же, однако конвертированию придается важное значение
4	То же, что и в случае 2, плюс наличие автоматизированных средств конвертирования и установки
5	То же, что и в случае 3, плюс наличие автоматизированных средств конвертирования и установки



Легкость эксплуатации

Значение	Описание
0	К эксплуатации не предъявляется никаких специальных требований, за исключением обычных процедур резервного копирования
1-4	Приложение обладает одной, несколькими или всеми из перечисленных далее возможностей. Каждая возможность, за исключением второй, обладает единичным весом: 1) наличие процедур запуска, копирования и восстановления с участием оператора; 2) то же, без участия оператора; 3) минимизируется необходимость в монтировании носителей для резервного копирования; 4) минимизируется необходимость в средствах подачи и укладки бумаги при печати
5	Вмешательство оператора требуется только при запуске и завершении работы системы. Обеспечивается автоматическое восстановление работоспособности приложения после сбоев и ошибок



Количество возможных установок на различных платформах

Значение	Описание
0	Приложение рассчитано на установку у одного пользователя
1	Приложение рассчитано на много установок для строго стандартной платформы (технические средства + программное обеспечение)
2	Приложение рассчитано на много установок для платформ с близкими характеристиками
3	Приложение рассчитано на много установок для различных платформ
4	То же, что в случаях 1 или 2, плюс наличие документации и планов поддержки всех установленных копий приложения
5	То же, что в случае 3, плюс наличие документации и планов поддержки всех установленных копий приложения



Простота изменений (гибкость)

Гибкость характеризуется наличием у приложения следующих возможностей:

- ▶ поддержка простых запросов, например, логики и (или) в применении только к одному внутреннему логическому файлу (ILF) (вес — 1)
- ▶ поддержка запросов средней сложности, например, логики и (или) в применении более чем к одному ILF (вес - 2)
- ▶ поддержка сложных запросов, например, комбинации логических связок и (или) в применении к одному или более ILF (вес — 3)
- ▶ управляющая информация хранится в таблицах, поддерживаемых пользователем в интерактивном режиме, однако эффект от ее изменений проявляется на следующий рабочий день
- ▶ то же, но эффект проявляется немедленно (вес — 2)

Значение	Описание
0	Ни одной из перечисленных функциональных возможностей
1	Любая одна из возможностей
2	Любые две возможности
3	Любые три возможности
4	Любые три возможности
5	Все пять возможностей

Пересчет FP-оценок в LOC-оценки

Язык программирования	Количество операторов на один FP
Ассемблер	320
C	128
Кобол	106
Фортран	106
Паскаль	90
C++	64
Java / C#	53
Ada 95	49
Visual Basic	32
Visual C++	34
Delphi Pascal	29
Perl	21
Prolog	54



Размер программного обеспечения в FP и LOC

Тип ПО	Размер, FP	Размер, LOC	Количество LOC на одну FP
Текстовые процессоры	3500	437500	125
Электронные таблицы	3500	437500	125
Клиент-серверные приложения	7500	675000	90
ПО баз данных	7500	937500	125
Производственные приложения	7500	937500	125
Крупные бизнес-приложения	10000	1050000	105
Операционные системы	75000	11250000	150
Системы масштаба предприятий	150000	18750000	125
Крупные оборонные системы	250000	25000000	100



Распределение временных затрат по стадиям для маленьких и больших проектов

Масштаб проекта	Начальная стадия, %	Проектирование, %	Разработка, %	Ввод в действие, %
Маленький коммерческий проект	10	20	50	20
Большой, сложный проект	15	30	40	15



Статистические данные по времени разработки

Размер проекта	< 100 FP	100 - 1000 FP	1000 - 10000 FP	> 10000 FP
Планируемый срок (мес.)	6	12	18	24
Реальный срок (мес.)	8	16	24	36
Отставание	2	4	6	12

Примечание: причины отставания частично объясняются неточной оценкой, частично — ростом количества требований к системе после того, как выполнена начальная оценка



Производительность разработчиков для стандартных типов проектов

Тип программы	Строк кода на человека-месяц (номинал)		
	проект на 1000 LOC	проект на 10000 LOC	проект на 25000 LOC
Авиационное оборудование	100 - 1000 (200)	20 - 300 (50)	20 - 200 (40)
Бизнес-система	800 - 18000 (3000)	200 - 7000 (600)	100 - 5000 (500)
Системы управления	200 - 3000 (500)	50 - 600 (100)	40 - 500 (80)
Встроенные системы	100 - 2000 (300)	30 - 500 (70)	20 - 400 (60)
Открытые Интернет-системы	600 - 10000 (1500)	100 - 2000 (300)	100 - 1500 (200)
Микрокод	100 - 800 (200)	20 - 200 (40)	20 - 100 (30)
Управление процессами	500 - 5000 (1000)	100 - 1000 (300)	80 - 900 (200)
Системы реального времени	100 - 1500 (200)	20 - 300 (50)	20 - 300 (40)
Системы научных и инженерных исследований	500 - 7500 (1000)	100 - 1500 (300)	80 - 1000 (200)
Коммерческие пакеты	400 - 5000 (1000)	100 - 1000 (200)	70 - 800 (200)
Системные программы/драйверы	200 - 5000 (600)	50 - 1000 (100)	40 - 800 (90)
Телекоммуникации	200 - 3000 (600)	50 - 600 (100)	40 - 500 (90)



Основные выводы

- ▶ Оценка трудоемкости должна быть вероятностным утверждением. Это означает, что для нее существует некоторое распределение вероятности, которое может быть очень широким, если неопределенность высокая, или достаточно узким, если неопределенность низкая
- ▶ Использование собственного опыта или опыта коллег, полученного в похожих проектах, это наиболее прагматичный подход, который позволяет получить достаточно реалистичные оценки трудоемкости и срока реализации программного проекта, быстро и без больших затрат
- ▶ Если собственный опыт аналогичных проектов отсутствует, а экспертное мнение недоступно, то необходимо использовать формальные методики, основанные на обобщенном отраслевом опыте, а именно метод функциональных точек или модель СОСМО II
- ▶ Недооценка приводит к ошибкам планирования и неэффективному взаимодействию. Агрессивные сроки, постоянное давление, сверхурочные, авралы служат причиной того, что затраты на проект растут экспоненциально и неограниченно





Спасибо за внимание!



Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана
Каф. ИУ-7

baryshnikovam@mail.ru

Лекция 10

Основы ценообразования на программные
продукты. Бюджет как основа планирования
деятельности



Возможные стратегии ценообразования на ПП

Основная проблема разработчика при оценке рыночной стоимости ПП сводится к определению цены, обеспечивающей максимальный доход фирмы



Много продаж
по малой цене

Мало продаж
по большой
цене

Большое
количество
пользователей

Относительно
высокая цена на
продукты

Низкая цена
продукта

Ограниченный
круг
пользователей



Типы рынков, влияющие на политику ценообразования ПП

- ▶ **Рынок чистой конкуренции:** рынок продавцов и покупателей, совершающих сделки со сходным программным продуктом в ситуации, когда ни один отдельный покупатель или продавец не оказывает большого влияния на уровень текущих цен, так как информация о ценах конкурентов общедоступна, а контроль цен отсутствует. На рынке чистой конкуренции роль маркетинговых исследований, политики цен, рекламы, политики сбыта и прочих мероприятий минимальна
- ▶ **Рынок монополистической конкуренции:** дифференцированный подход к ценообразованию программного продукта предполагает, что цены на него могут устанавливаться в широком диапазоне. Наличие диапазона цен объясняется способностью производителей предложить потребителям разные варианты услуг по продаже и сопровождению ПП, включая сопутствующие программные продукты и дополнительные сервисы по поддержке пользователя. Чтобы выделиться чем-то помимо цены на ПП и перечня услуг разработчики ориентируются на разные потребительские сегменты, широко используют рекламу и другие каналы продвижения продукции



Типы рынков, влияющие на политику ценообразования ПП

- ▶ **Олигополистический рынок** предполагает доминирование небольшого числа производителей, чувствительных к политике ценообразования и маркетинговой стратегии друг друга. Новым претендентам тяжело проникнуть на этот рынок: если какая-то компания снизит цены на свои продукты, то другим производителям придется либо тоже снижать цены, либо предлагать дополнительные сервисы
- ▶ **Рынок чистой монополии:** на таком рынке присутствует всего один продавец. В случае регулируемой монополии государство может устанавливать для такого продукта цену, обеспечивающую получение «справедливой нормы прибыли», позволяющую организации не только поддерживать производство, но и при необходимости расширять его. При нерегулируемой монополии фирма сама вправе устанавливать любую цену, которую выдержит рынок



Альтернативные стратегии ценообразования

- ▶ **Обеспечение выживаемости ПП на рынках.** Проблемы могут быть вызваны активизацией конкурентов или изменившимися запросами потребителей. Понимая, что в какой-то момент времени выживание для фирмы важнее прибыли, она может установить низкие цены на свои продукты в надежде на благоприятную реакцию потребителей. До тех пор, пока снижение цены на программный продукт покрывает издержки по его разработке продукт следует продвигать
- ▶ **Максимизация прибыли.** При реализации данной стратегии фирмы ориентируются на уровень цен, обеспечивающих в краткосрочном периоде получение максимальной прибыли, и не рассматривают долгосрочные перспективы
- ▶ **Максимальное увеличение объема продаж.** Фирма снижает цены на свою продукцию до минимально допустимого уровня, повышая долю своего рынка, добивается снижения издержек на единицу продукции и на этой основе может и дальше снижать цены. Эта политика хорошо срабатывает, если чувствительность рынка к ценам велика

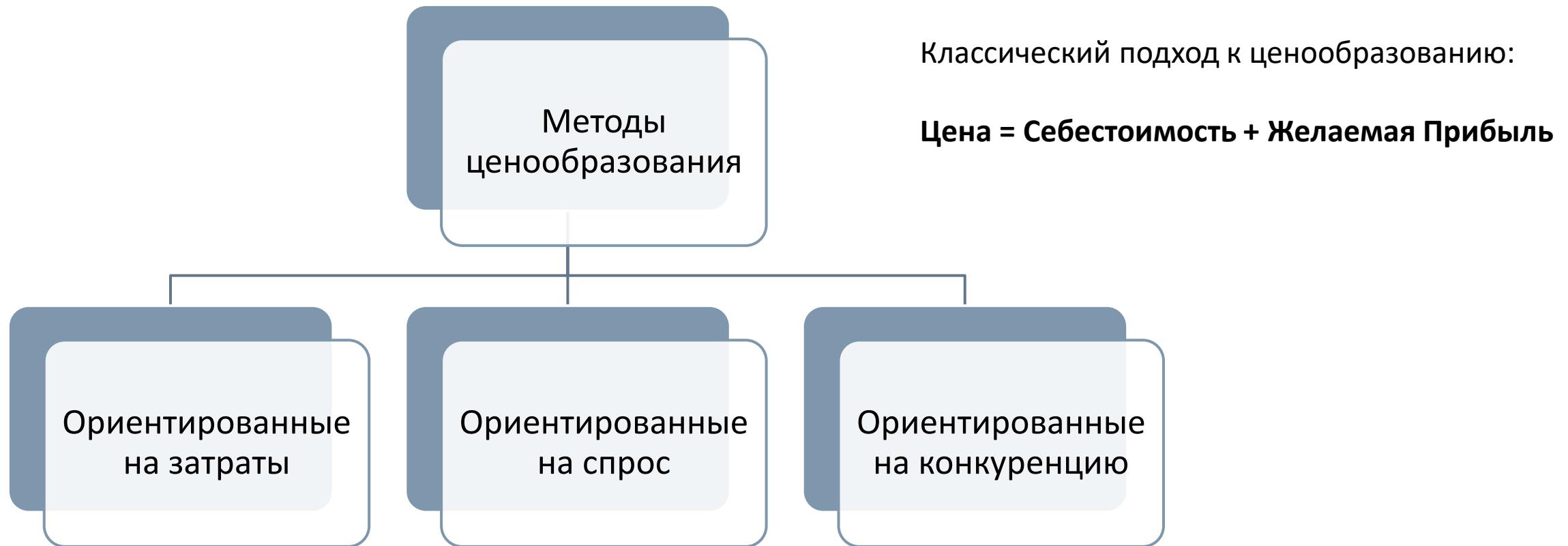


Альтернативные стратегии ценообразования

- ▶ **«Снятие сливок» за счет установления высоких цен.** Фирма устанавливает на новый продукт, который выходит на рынок максимально возможную цену благодаря сравнительным преимуществам новой разработки. Как только продажи продукта по данной цене сокращаются, фирма снижает цену, привлекая к себе следующий слой потребителей, достигая в каждом сегменте целевого рынка максимально высокого оборота
- ▶ **Достижение лидерства по качеству.** Фирма, которая способна закрепить за собой репутацию лидера по качеству, устанавливает высокую цену, чтобы покрыть значительные издержки, связанные с обеспечением высокого качества продукта



Методы ценообразования



Минимальная цена определяется себестоимостью, а максимальная – наличием каких-то уникальных потребительских свойств продукта или монопольным положением фирмы в соответствующем сегменте рынка



Методы ценообразования, ориентированные на затраты

Основаны на учете расходов производителя по созданию и реализации продукта. При определении цены необходимо выявить диапазон цен, обеспечивающих покрытие переменных и постоянных затрат и получение желаемой прибыли

Производство тиражного продукта	Производство ПО на заказ
Затраты на разработку и тестирование первой версии ПП	Себестоимость первой версии ПП
Затраты на распространение	Затраты на сопровождение
Затраты на обслуживание	Затраты на создание новых версий, отвечающих изменившимся потребностям заказчика
Затраты на модернизацию	
Затраты на поддержку пользователей в течение всего периода эксплуатации ПП	



Методы ценообразования, ориентированные на спрос

- ▶ Предусматривают готовность потребителей оплачивать программные продукты по верхней границе цены
- ▶ При использовании данных методов не прослеживается связь между затратами и ценой, кроме того, что цена не может быть ниже уровня нижней границы
- ▶ Оценка спроса имеет прогнозный характер и его труднее выразить в количественных величинах, чем издержки
- ▶ Задача по оценке спроса усложняется, если на рынок выводится новый продукт в силу отсутствия статистики за прошлые периоды

Рекомендации по минимизации рисков прогнозирования спроса: лучше первым завершить разработку и захватить рынок, даже если это приведет к увеличению затрат



Методы ценообразования, ориентированные на конкурентов

- ▶ Основываются на сравнении потребительских свойств ПП разработчика и конкурентов
- ▶ Потребительская ценность программного продукта зависит от ряда факторов, таких как качество продукта, возможность подключения других приложений, платформонезависимость и др.
- ▶ При использовании данной политики существуют две проблемы:
 - у потенциальных пользователей нет навыков объективного анализа потребительских свойств ПП
 - точка зрения пользователя и точка зрения разработчика на потребительские свойства ПП может различаться
- ▶ Потребители как правило отдают предпочтение продуктам, позволяющим получить максимальную выгоду при их использовании, определяемую как разность между потребительской ценностью и общей стоимостью продукта, включающей единовременные затраты на приобретение и внедрение и текущие расходы на эксплуатацию
- ▶ При сравнении ПП с продуктами конкурентов не обязательно, чтобы он превосходил их по всем параметрам. Необходимо выбрать главное конкурентное преимущество и на нем делать акцент



Корректировка базовой цены на ПП

Базовая цена, установленная по любому из перечисленных методов ценообразования, может быть скорректирована в сторону уменьшения за счет:

- ▶ установления скидок при большом заказе количества лицензий
- ▶ предложения различных форм оплаты: с предоплатой, оплатой по факту получения продукта, оплаты с рассрочкой и т.д.
- ▶ установления скидок определенным группам клиентов
- ▶ включением в цену продажи сервисов по техническому сопровождению и поддержке пользователей
- ▶ предоставления возможности выбора набора функциональности ПП и способа поставки (стандартная, профессиональная, промышленная версии)



Бюджет

Бюджет - это выраженный в деньгах количественный план реализации проекта, подготовленный и принятый на определенный период времени и показывающий планируемую величину **дохода**, его источники и **расходы**, предстоящие в течение этого периода, а также в случае необходимости – **величину привлекаемого капитала**

Назначение бюджета:

- ▶ контроль производственной ситуации (в отсутствие бюджета руководитель может только констатировать текущее состояние проекта, вместо того, чтобы реагировать на возникающие проблемы)
- ▶ объективная оценка результатов деятельности как в целом, так и по отдельным направлениям реализации, за счет сравнения плановых и фактических показателей
- ▶ выявление за счет сравнения плановых показателей бюджета и фактических результатов тех ситуаций, которые требуют корректирующего воздействия



Основные показатели, используемые для оценки стоимости ПП

- ▶ Сложность (размер)
- ▶ Трудозатраты на разработку
- ▶ Длительность разработки продукта в целом и ее отдельных этапов
- ▶ Численность и квалификация специалистов, привлекаемых к созданию программного продукта
- ▶ Размеры фондов оплаты труда специалистов на создание ПП в целом и по каждому этапу жизненного цикла
- ▶ Прочие прямые затраты и накладные расходы, связанные с созданием ПП

Фонд оплаты труда на реализацию проекта определяется исходя из производительности труда специалистов и согласованной базовой ставки заработной платы (руб./час). Остальные показатели можно рассчитать на основе метода функциональных точек и моделей СОСОМО и СОСОМО II



Этапы составления бюджета

- ▶ Определение основных параметров бюджета
- ▶ Планирование доходной части
- ▶ Планирование бюджета основных расходов
- ▶ Планирование бюджета накладных расходов

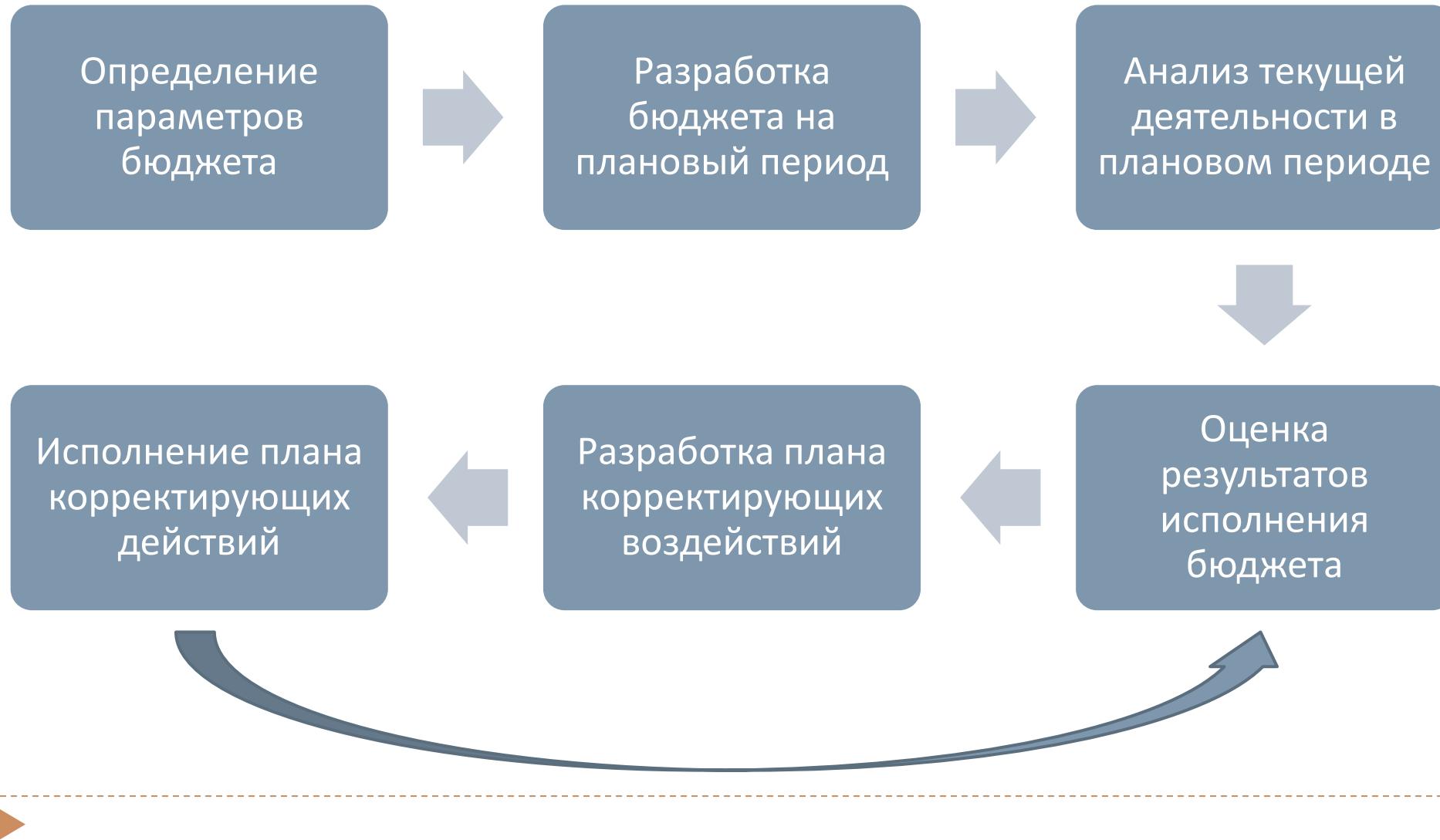
Процесс составления бюджета и контроля за его исполнением называется бюджетным циклом

Нормативы, используемые при разработке бюджета:

- ▶ Рыночная цена продажи одной лицензии на ПП
- ▶ Ожидаемый объем продаж ПП
- ▶ Ожидаемые трудозатраты реализации проекта по разработке ПП в целом и его отдельных этапов
- ▶ Расчетная ставка стоимости одного часа труда специалиста, занятого в реализации проекта
- ▶ Норматив отчисления на накладные расходы
- ▶ Нормы расхода материальных ресурсов на единицу стоимости ПП



Взаимосвязь этапов бюджетного цикла



Бюджет затрат на разработку проекта

№	Наименование статей расходов	Плановые периоды			
		1 кв.	2 кв.	3 кв.	4 кв.
1	Фонд оплаты труда исполнителей (ФОТ)				
2	Начисления на ФОТ				
3	Материальные затраты				
4	Увеличение стоимости основных средств				
5	Командировочные расходы				
6	Оплата услуг сторонних организаций				
7	Прочие расходы				
8	Амортизация				
9	Накладные расходы				
10	Налог на добавленную стоимость				
11	Планируемая прибыль				



Затраты и их классификация

Затраты отражают все расходы организации, связанные с получением основных и прочих доходов

В зависимости от объемов производства

Постоянные (не зависят от изменения объемов производства)

Переменные (изменяются вместе с изменением объемов производства)

В зависимости от роли в реализации основных видов деятельности

Основные (непосредственно связаны с технологическим процессом изготовления продукции)

Накладные расходы (затраты на организацию и обслуживание производства, реализацию продукции, управление организацией)

В зависимости от времени возникновения и отнесения на себестоимость продукции

Текущие затраты (затраты по производству и реализации продукции текущего отчетного периода)

Расходы будущих периодов (произведенные затраты текущего отчетного периода, которые будут отнесены на себестоимость продукции, которая будет выпущена в последующие периоды)



Классификация затрат по статьям калькуляции

- ▶ заработка плата работников
- ▶ отчисления на социальные нужды
- ▶ увеличение стоимости основных фондов (приобретение компьютерной и офисной техники)
- ▶ приобретение расходных материалов и комплектующих
- ▶ затраты на подготовку и переподготовку персонала
- ▶ налог на прибыль
- ▶ оплата коммунальных услуг
- ▶ оплата услуг связи
- ▶ командировочные расходы

Планирование и учет затрат по статьям калькуляции позволяет группировать расходы по целевому назначению: непосредственно связанные с производством продукции, затраты на управление предприятием, на реализацию продукции



Себестоимость продукции

Себестоимость – это стоимостная оценка используемых в процессе выполнения работ (оказания услуг) материальных и трудовых ресурсов, основных фондов, энергии, а также других затрат на производство и реализацию

Затраты, образующие себестоимость продукции:

- ▶ Материальные затраты
- ▶ Затраты на оплату труда
- ▶ Отчисления на социальные нужды
- ▶ Амортизация основных фондов
- ▶ Прочие затраты



Материальные затраты

- ▶ Стоимость приобретенных на стороне компонентов, которые входят в разрабатываемые программные продукты
- ▶ Стоимость покупных материалов, используемых для обеспечения нормального технологического процесса производства (бумага, картриджи, сменные носители и др.) и реализации продукции (например, внешнего носителя, на который записывается дистрибутив, или упаковки), а также технической литературы (при необходимости)
- ▶ Стоимость работ (услуг) производственного характера, выполняемых сторонними организациями
- ▶ Коммунальные платежи: отопление, электроэнергия...



Затраты на оплату труда

- ▶ Выплаты заработной платы за фактически выполненную работу
- ▶ Выплаты стимулирующего характера, премии и другие вознаграждения по итогам работы
- ▶ Стоимость выдаваемых бесплатно предметов, остающихся в постоянном пользовании
- ▶ Оплата очередных и дополнительных отпусков, компенсация за неиспользуемый отпуск
- ▶ Выплаты работникам, высвобождаемым с предприятий по сокращению штатов
- ▶ Оплата учебных отпусков, предоставляемых работникам, обучающимся в вечерних и заочных учебных заведениях, заочной аспирантуре
- ▶ Оплата труда работников, не состоящих в штате организации, за выполнении ими работ по заключенным договорам гражданско-правового характера



Группы специалистов, принимающие участие в разработке программного продукта

- ▶ Руководитель проекта, линейные менеджеры, системные аналитики
- ▶ Непосредственные разработчики программных систем и специалисты по интеграции
- ▶ Технический персонал, обеспечивающий тестирование, документирование и опытную эксплуатацию ПП



Определение фонда оплаты труда на разработку программного продукта

Производится на основании показателя производительности труда программиста, который выражается в количестве строк программного кода, написанного за один рабочий день

Нормативы трудоемкости разработки программ

Уровень сложности программного продукта	Размер программного продукта	
	Простые (до 30 тыс. LOC)	Сложные (до 500 тыс. LOC)
1 тип (преимущественно на языке Ассемблер)	до 14 строк/человеко-день	до 8 строк/человеко-день
2 тип (на языках высокого уровня)	до 22 строк/человеко-день	до 16 строк/человеко-день

Примечание: данные нормативы отражают не только трудоемкость непосредственного написания текстов программ, но и процессы интеграции и тестирования всего программного комплекса



Определение фонда оплаты труда

В качестве основания для определения фонда оплаты труда используются:

- ▶ длительность реализации каждого этапа жизненного цикла разработки программного продукта
- ▶ количественный и качественный состав специалистов, привлекаемых на каждом этапе разработки
- ▶ базовая месячная ставка специалиста-программиста

Трудозатраты на разработку ПП:

$$T = R / P,$$

где

R – размер программного кода, LOC

P – производительность труда
специалистов

Средняя численность специалистов:

$$Z = T / D,$$

где

D – длительность реализации проекта

Примечание: Для вычисления Т могут быть использованы методики СОСМО и СОСМО II



Распределение трудозатрат и длительности разработки по отдельным этапам жизненного цикла

Этапы жизненного цикла	Трудозатраты, α , %	Длительность разработки, β , %
1. Анализ предметной области и разработка требований	8 - 10	10 - 20
2. Проектирование	14 - 22	20 - 30
3. Программирование	41 - 46	32 - 35
4. Тестирование и комплексные испытания	24 - 27	2 - 25

Средняя численность сотрудников, занятых на каждом из этапов жизненного цикла создания ПП, где $i=1..4$

$$Z_i = \frac{\alpha_i * T}{\beta_i * D}$$



Распределение специалистов по этапам жизненного цикла ПП

Этапы жизненного цикла	Типы специалистов, %		
	аналитики	программисты	технические специалисты
1. Анализ предметной области и разработка требований	40	20	40
2. Проектирование	35	35	30
3. Программирование	10	65	25
4. Тестирование и комплексные испытания	15	60	25

Численность каждого типа специалистов на каждом из этапов жизненного цикла создания ПП

$Z_{ij} = P_{ij} * Z_i$, где $i=1..4, j=1..3$, где P_{ij} – относительная доля специалистов j -го типа, привлекаемых для реализации проекта на i -м этапе, %



Определение фонда заработной платы

Фонд заработной платы для реализации i-го этапа проекта:

$$Z_i = \sum_{j=1}^3 Z_{ij} * D_i * S_j$$

D_i – длительность i-го этапа проекта,
S_j – месячный фонд заработной платы
специалиста j-го типа

Примечание: В основу определения S_i может быть положена базовая ставка программиста на фирме – разработчике ПП, либо рыночная базовая ставка программиста в данном регионе

Соотношение месячной ставки программиста к месячной ставке системного аналитика составляет 1:1,3, а к месячной ставке технического специалиста 1:0,7

Общий фонд заработной
платы на реализацию проекта
составляет:



Отчисления на социальные нужды

Обязательные отчисления по установленным законодательством нормам в:

- ▶ органы государственного страхования,
- ▶ пенсионный фонд,
- ▶ государственный фонд занятости населения
- ▶ фонд медицинского страхования

Амортизация основных фондов

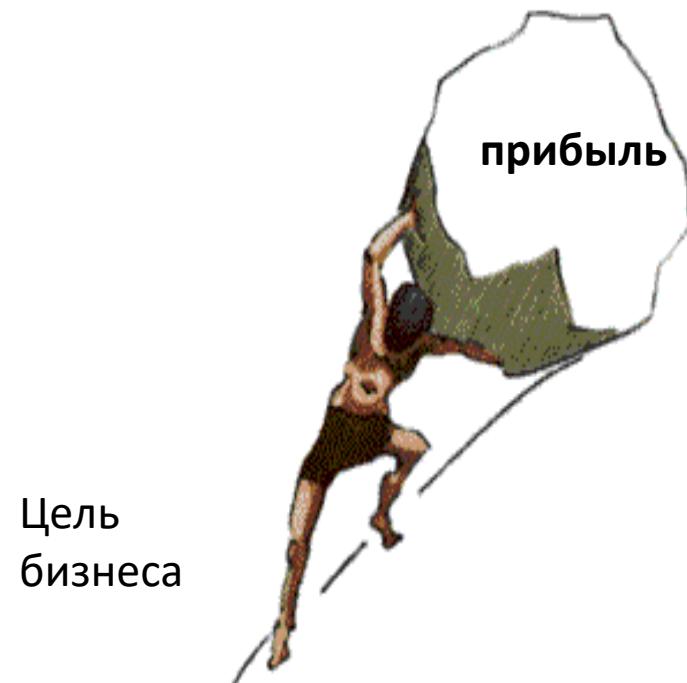
Отражает сумму амортизационных отчислений на полное восстановления основных средств, исчисленную исходя из их балансовой стоимости

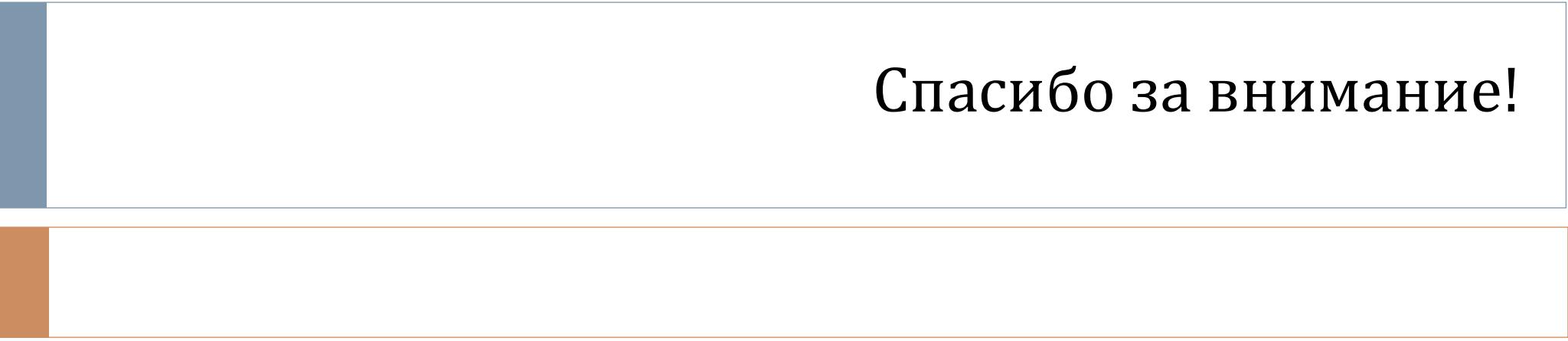
Прочие затраты

Включают расходы на канцелярские товары, командировки, оплату телефонных переговоров, мобильной связи, подписки на СМИ и ПО, представительские расходы и пр.



Схема формирования показателей прибыли





Спасибо за внимание!

Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана

baryshnikovam@mail.ru

Лекция_11. Вместо заключения

Способы повышения эффективности
программных проектов или о чем говорит
актуальная статистика их реализации.
Информация к размышлению

Рекомендации компании Standish Group

- Посчитайте то, что поддается счету
- Улучшайте клиента и пользователя
- Увеличивайте стоимость проекта
- Снижайте затраты на проект
- Улучшайте среду проекта

Источник: *CHAOS Report 2015*



Основные риски программных проектов

1. Неясные и неточные цели проекта (степень важности близка к 100%);
2. Неадекватные сроки реализации проекта и неадекватный бюджет (степень важности порядка 75%);
3. Непродуманные и плохо отлаженные коммуникации между заинтересованными лицами проекта (степень важности составляет примерно 79%);
4. Убытки из-за отсутствия коммуникаций с пользователями (степень важности порядка 83%);
5. Не отвечающие требованиям проекта знания и умения (степень важности примерно 57%);
6. Отсутствие эффективной методологии управления проектом (степень важности близка к 60%);
7. Недооценка требований проекта (степень важности оценивается в 92%);
8. «Золотое покрытие» (gold plating) – завышение аналитиком и / или менеджером требований проекта с целью сделать ПП лучше и удобнее (степень важности незначительная -12%);
9. Изменение требований в процессе разработки проекта (степень важности составляет 63%);
10. Ошибки (bag), допускаемые в процессе разработки (степень важности близка к 90%);
11. Субподрядчики;
12. Низкая производительность проектной команды;
13. Применение новых технологий в проекте;
14. Некачественное управление ожиданиями

Источник: Addison T., Vallabh S. Controlling Software Project Risks – an Empirical Study of Methods used by Experienced Project Managers

Оценка успешности IT-проектов на основе традиционного подхода (OnTime, OnBudget, OnTarget)

	2011	2012	2013	2014	2015
успешные	39%	37%	41%	36%	36%
провальные	22%	17%	19%	17%	19%
спорные	39%	46%	40%	47%	45%

Успешные проекты – все цели проекта достигнуты в плановый срок и бюджет

Провальные проекты – проекты, остановленные без получения результата

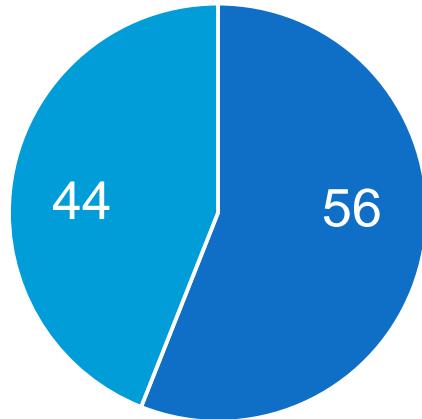
Спорные проекты – те проекты, которые были завершены либо с превышением сроков, либо стоили дороже, чем планировалось, либо достигли лишь части целей

Примечание: Общее количество проанализированных программных проектов за 2011-2015 более 25 000, в среднем 5 000 в год



Декомпозиция результатов ИТ-проектов, реализованных в 2011 – 2015 гг.

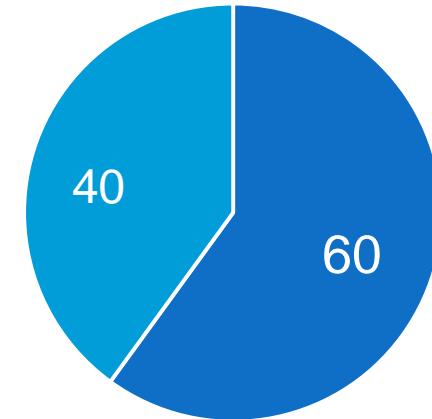
OnBudget



■ нет ■ да

Процент проектов, которые уложились в бюджет

OnTime



■ нет ■ да

Процент проектов, которые были выполнены в срок



Оценка успешности ИТ-проектов на основе современного подхода (OnTime, OnBudget и удовлетворенность клиентов)

	2011	2012	2013	2014	2015
успешные	29%	27%	31%	28%	29%
провальные	22%	17%	19%	17%	19%
спорные	49%	56%	50%	55%	52%

Степень удовлетворения клиента существенно выше, когда реализованные в рамках проекта функции не просто автоматически соответствуют запланированным, а отвечают именно его реальным потребностям. Автоматическое следование ТЗ часто приводит к тому, что реализованные функции программного обеспечения не используются, но увеличивают стоимость и степень риска проекта и сказываются на качестве: масштаб проекта растет, но не обязательно обеспечивается ценность для пользователя

Примечание: Произошло перераспределение между успешными и спорными проектами



Оценка степени влияния размера проекта на его результат

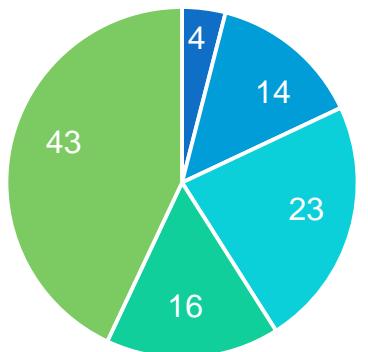
Размер	Результат		
	успешный	спорный	провальный
Очень большой	6%	51%	43%
Большой	11%	59%	30%
Средний	12%	62%	26%
Умеренный	24%	64%	12%
Маленький	61%	32%	7%

Примечание: Т.е. только 6% из всех очень больших проектов оказались успешными, тогда как для маленьких проектов процент успеха составил 61%



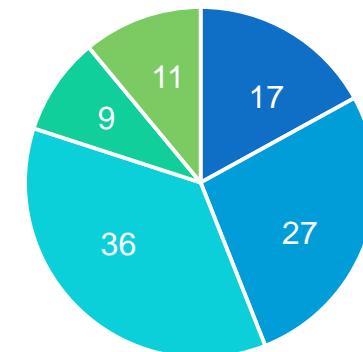
Финансовая отдача от реализации программных проектов

Ценность ("отдача") для организации от реализации больших проектов



- Очень высокая ценность
- Высокая ценность
- Средняя
- Низкая
- Очень низкая

Ценность ("отдача") для организации от реализации маленьких проектов



- Очень высокая ценность
- Высокая ценность
- Средняя
- Низкая
- Очень низкая

Примечание: Т.е. практически 60% больших проектов не принесли ценности той организации, которая их реализовывала

Примечание: Только 20% маленьких проектов не приносят ценности той организации, которая их реализует



Выводы и рекомендации

- ▶ Чем крупнее проект, тем не только меньше вероятность достижения успеха, но и тем скромнее его норма доходности (норма прибыли)
 - ▶ Во многих случаях крупные проекты никогда не окупают затраты организации, которая их реализует
 - ▶ Чем быстрее проекты идут в производство, тем быстрее они начинают окупаться
-
- ▶ Целесообразно разбивать большой проект по созданию программного обеспечения на несколько небольших проектов, с ранней доставкой результатов до потребителя, с целью более быстрой финансовой отдачи и большей удовлетворенности пользователей
 - ▶ Большинство программных проектов требуют небольшой команды, которая привлекается на короткий срок для того, чтобы сформировать ценность для организации.

Только в очень редких случаях проекты должны быть большими и длинными



Распределение программных проектов по отраслям экономики

Отрасли экономики	успешные	спорные	провальные
Банковское дело	30%	55%	15%
Финансы	29%	56%	15%
Государственные контракты	21%	55%	24%
Здравоохранение	29%	53%	18%
Производство	28%	53%	19%
Торговля	35%	49%	16%
Сервис / Сфера услуг	29%	52%	19%
Телекоммуникации	24%	53%	23%
Другие	29%	48%	23%

Примечание: Наиболее успешными были проекты в сфере розничной торговли (Retail): их показатель успеха 35%. Самый высокий уровень неудач (24%) показали проекты, реализуемые в рамках государственных заказов. Результаты проектов в финансовой сфере оказались наиболее спорными (56%)



Уровень удовлетворенности клиентов результатами программных проектов



Географическое распределение результатов программных проектов

География реализации проекта	успешные	спорные	провальные
Северная Америка	31%	51%	18%
Европа	25%	56%	19%
Азия	22%	58%	20%
Остальной мир	24%	55%	21%

Примечание: Одной из причин успешности американских проектов является наличие у менеджеров проектных команд эмоциональной зрелости – т.е. навыков, ориентированных на управление ожиданиями клиентов и достижение консенсуса всех заинтересованных сторон, что приводит к высокому уровню удовлетворенности



Анализ зависимости результата от типа проекта

Тип проекта	успешные	спорные	провальные
Разработка с нуля с использованием традиционных языков и методов	22%	61%	17%
Разработка с нуля с использованием современных методов	23%	54%	23%
Разработка с использованием готовых и/или покупных компонентов	24%	59%	17%
Приложение собрано из покупных компонентов	25%	59%	16%
Модификация покупных приложений	42%	37%	21%
Покупка и использование готовых приложений (коробочные продукты)	57%	28%	15%
Модернизация	53%	38%	9%
Другие	28%	47%	25%

Примечание: Самый высокий показатель успешности был у коробочных продуктов (57%). Проекты, которые разрабатывались с нуля с использованием традиционных языков и методов, показали самый высокий уровень спорности результатов – 61%.



Влияние метода разработки на результат проекта

Тип проекта	метод	успешные	спорные	провальные
Проекты любого размера	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Проекты большого размера	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Проекты среднего размера	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Проекты маленького размера	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

Примечания: Agile проекты оказались почти в четыре раза более успешными, чем проекты, реализуемые по методу водопада. Водопадные проекты не очень хорошо масштабируются в отличие от гибких. Для маленьких проектов характерна небольшая разница в уровне успешности между гибким и водопадным процессами.

Вывод: Самое выгодное сочетание – это гибкий процесс и небольшие проекты, они имеют только 4% неудач



Соответствие цели или удовлетворенность потребителя?

Степень соответствия проекта первоначальной цели	успешные	спорные	провальные
Точное	22%	53%	25%
Примерное	23%	54%	23%
Свободное	27%	52%	21%
Расплывчатое (неопределенное)	38%	46%	16%
Очень отдаленное	34%	58%	8%

Основные выводы: Бизнес-цели практически любого проекта динамичны, в том числе, их может сильно изменить и ход его реализации. Многие из наиболее успешных проектов начинались как неопределенные.

Слишком жесткое следование изначальной стратегии может препятствовать достижению высоких показателей удовлетворенности клиентов и, как следствие, - успеху. Проектные группы должны сбалансировать усилия по контролю бизнес-целей, чтобы поощрять продвижение инноваций



Влияние уровня компетентности персонала на результаты проектов

Качество персонала в команде проекта	успешные	спорные	провальные
Одаренный	38%	45%	17%
Талантливый	31%	53%	16%
Компетентный/квалифицированный	28%	53%	19%
Способный	24%	54%	22%
Неподготовленный/низкой квалификации	17%	60%	23%

Пять основных принципов обеспечения компетентности персонала:

- определить необходимые компетенции и альтернативные навыки
- обеспечить хорошую, непрерывную программу обучения для повышения квалификации персонала
- нанимать как внутренних, так и внешних сотрудников для обеспечения баланса опыта
- обеспечить стимулы для мотивации персонала
- убедиться, что персонал ориентирован на проект

Вывод: Когда проект обеспечен людьми, имеющими соответствующий уровень профессиональной компетентности и обладающими навыками командной работы, он может рассчитывать на успех даже в самых тяжелых условиях реализации



Факторы, влияющие на успех проекта

Фактор	Баллы
Заинтересованность менеджера проекта и поддержка менеджмента организации	15
Вовлеченность пользователя в процесс разработки	15
Оптимизация, т.е. декомпозиция большого проекта на несколько малых	15
Эмоциональная зрелость: управление ожиданиями и достижение консенсуса	15
Улучшение профессиональных навыков	10
Agile процессы, т.е. отказ от классической разработки (waterfall) в пользу гибких методологий реализации проекта	7
Ясные цели проекта	4
Готовность привлекать внешнюю экспертизу	6
Опыт управления проектами	5
Инструменты и инфраструктура проекта	8



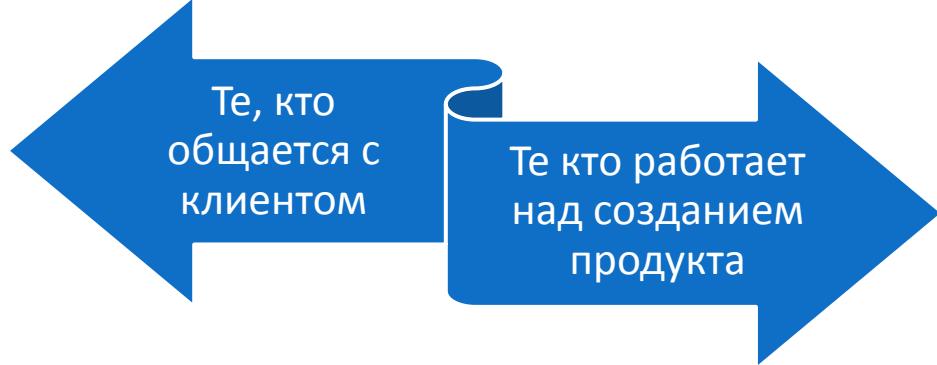
Рекомендации по повышению эффективности проектов

- ▶ Создание документа, в котором заказчиком коротко и ясно формулируются цели, задачи проекта, описываются пользователи и предполагаемый продукт
- ▶ Привлечение независимых экспертов для идентификации и оценки рисков
- ▶ Декомпозиция большого проекта на несколько малых проектов, т.е. создание портфеля проектов (project portfolio) с назначением менеджера портфеля проектов, отвечающего за успешное завершение данных проектов
- ▶ Наличие предварительных грубых оценок: погрешность оценки срока реализации ИТ-проекта и бюджета на этапе инициации не должна превышать 50%, а на этапе на этапе планирования - 10–15%
- ▶ Применение современных гибких методологий в процессе разработки ИТ-проекта
- ▶ Создание фокус-групп из потенциальных пользователей с целью тестирования инкрементов продукта в процессе реализации ИТ-проекта



Навыки, которые нужны ИТ-специалисту

В традиционных подходах к реализации IT-проектов всегда существовало чёткое разграничение между людьми, которые управляют проектом, взаимодействуют с бизнесом, и разработчиками, которые просто выполняют поставленную задачу



Источник:

<https://tproger.ru/articles/10-software-skills-for-it-specialist/>

В современных реалиях эта модель не всегда применима и очень часто не успешна

Основные причины: очень часто итоговый результат отличается от того, что на самом деле было нужно клиенту или пользователям, плюс срыв сроков и превышение бюджета

Как было сказано выше при разработке ПО все чаще используются гибкие методологии (Agile), что привело к размытию границ между ролями и увеличению объемов коммуникации с бизнесом и внутри проектной команды



10 soft skills, которые нужны ИТ-специалисту

1. Понимание ценности для клиента	6. Переговорные навыки
2. Навыки коммуникации	7. Гибкость и креативность
3. Эмоциональный интеллект	8. Проактивность
4. Командная работа	9. Навыки презентации
5. Тайм-менеджмент	10. Менторинг

«Существует заблуждение, что гибкие навыки нужны в ИТ только тем, кто хочет стать менеджером. Нет, гибкие навыки пригодятся любому. Благодаря им, вы сможете легко доносить свои мысли до коллег, грамотно объяснять свои действия и логику. Сотруднику с хорошими «софт скиллами» доверяют интересные сложные проекты, коммуникацию с заказчиками и много дополнительных задач, которые разнообразят его рабочий день и делают работу увлекательней»



Понимание ценности для клиента

- ▶ В современном мире ИТ разработчику нужно понимать бизнес заказчика. Только так можно сделать по-настоящему хороший продукт
- ▶ Наличие этого навыка повышает уровень доверия между разработчиком и заказчиком. Если разработчик действительно понимает, зачем он делает то, что делает, он сможет выбрать правильный подход и решение и вовремя подскажет, что задачу можно выполнить иначе (либо не делать вообще)
- ▶ Понимание ценности для клиента позволит избежать переделок. Если разработанный продукт в итоге не решает проблему заказчика или не интересен пользователям, его придётся переделывать (клиент потратит больше денег и засомневается в уровне вашей компетентности)

Как сформировать навык: Всегда вначале думать о проблеме, а потом уже о задаче. Часто разработчик начинает выполнять задачу, не задумываясь о том, какую проблему решает. Возможно, задача не решает никакой проблемы или решает, но не ту



Навыки коммуникации

- ▶ Коммуникация - один из базовых социально-психологических навыков, который пригодится в любой сфере жизни и работы. Это умение слушать и слышать, убеждать и аргументировать, вести переговоры и организовывать работу команды, поддерживать отношения и работать в команде
- ▶ Коммуникация – один из важнейших навыков, необходимых для взаимодействия с командой проекта. Каким бы профессионалом вы себя ни ощущали, всегда найдётся задача, которую вы не в состоянии решить в одиночку
- ▶ Навыки коммуникации нужны для обмена опытом и знаниями. Если тем, что умеете вы, будут владеть и другие члены команды, то от этого выиграют все. Это работает и в обратную сторону — если в вашей команде есть человек, обладающий какими-то уникальными навыками, он может поделиться этим с вами. Это бесконечный цикл, в котором выигрывает каждый: команда, проект и, конечно же, каждый член команды в отдельности

Как сформировать навык: Будьте открытыми и идите навстречу общению. Помогайте другим и не бойтесь просить о помощи. Ну и конечно больше общайтесь! Только так можно сформировать этот навык, никакие курсы и книги тут не помогут



Эмоциональный интеллект

- ▶ Под эмоциональным интеллектом понимают эмпатию, способность видеть и распознавать скрытые эмоции людей, навык управления собственными эмоциями при принятии практических решений:
- ▶ Первая сторона эмоционального интеллекта — это понимание чувств и эмоций других людей. Нужно учиться понимать чувства и мотивацию своих коллег, заказчиков и партнеров, уметь с ними взаимодействовать
- ▶ Вторая сторона эмоционального интеллекта — это умение управлять своими эмоциями. Часто мы совершаем спонтанные действия под влиянием эмоций, например от злости или раздражения. Если мы понимаем свои эмоции и контролируем их, то можем чаще принимать осознанные и взвешенные решения

Как сформировать навык: Для понимания других людей учитесь навыку **активного слушания**, будьте вовлечеными в диалог, задавайте вопросы.

Старайтесь анализировать своё и чужое поведение **объективно**.

Don't be toxic — не позволяйте себе токсичного поведения. Не критикуйте и не жалуйтесь, особенно публично, потому что подобные действия деструктивны — они не несут пользы вам и окружающим, не создают ценности, а только ухудшают настроение и микроклимат в коллективе



Командная работа

- ▶ Как бы хороши вы ни были, есть задачи и проекты, с которыми вы не можете справиться в одиночку, не потратив на это кучу времени. Есть проекты, на которых задействовано от 100 до 1000 человек и даже больше. И вовсе не потому, что нужно создать много рабочих мест, а потому, что проект действительно масштабный, и с меньшим количеством участников его не реализовать
- ▶ Каждый человек обладает уникальным набором навыков, и только собрав вместе таких людей, можно получить хороший результат

Как сформировать навык: Решайте совместно с кем угодно, какие угодно задачи — будь то студенческий проект, домашние дела или что-то связанное с хобби — главное, делать это не одному



Тайм-менеджмент

- ▶ Вопреки общему заблуждению, мы не можем управлять временем. Его у нас ровно 24 часа каждый день
- ▶ Искусство грамотно управлять своим временем — один из самых важных навыков, так как помимо основных задач на проекте, у вас постоянно появляются новые. При неверном планировании работа становится непродуктивной
- ▶ Тайм-менеджмент — это не про время, а про задачи и приоритеты. О том, как управлять делами, чтобы сохранять продуктивность в периоды высокой нагрузки, избавлять себя от отвлекающих факторов и соблюдать баланс между работой и жизнью

Как сформировать навык:

- Держать все задачи в одном месте. Сегодня мы получаем информацию отовсюду — почта, мессенджер, звонки, что-то устно, что-то записано в блокноте или на стикерах, это затрудняет работу, поэтому важно держать всё в одном месте — будь то блокнот или мобильное приложение
- Разбивать большие задачи на мелкие, вплоть до таких, как «сходить», «написать», «позвонить». Человеческий мозг боится больших задач, потому что не понимает, с чего начинать, и мы начинаем откладывать задачу снова и снова, до тех пор, пока не придёт дедлайн
- Планировать, что за чем делать, когда большая задача разбивается на маленькие



Переговорные навыки

- ▶ Во-первых, это нужно, чтобы «продавать» свои идеи и навыки
- ▶ Во-вторых, важно уметь убеждать собеседников. Переговорные навыки необходимы даже на кадровом собеседовании. Каждое собеседование — это переговоры: у вас есть своя позиция и условия, у работодателя — свои. Каждая сторона защищает свои потребности и пытается найти компромисс

Как сформировать навык:

- Первое — готовьтесь к переговорам. Они пройдут хорошо, если не будут для вас неожиданностью. Найдите информацию о собеседнике, подумайте, чего хотите достичь — поставьте цель
- Практикуйте переговоры в повседневной жизни



Гибкость и креативность

- ▶ Любуому разработчику нужно быть готовым к изменениям и уметь принимать решения в нестабильной ситуации. Именно гибкость ума, креативность и нестандартное мышление — это главные принципы Agile-подхода

Как сформировать навык:

- старайтесь избегать думать стандартно, по шаблону, руководствоваться стереотипами. Часто мы сталкиваемся с ситуацией, когда на наши предложения мы получаем ответ «в этой организации так не принято делать». Остановитесь и подумайте, почему здесь так заведено и что будет, если вы сделаете иначе, может быть, получится что-то новое и интересное
- Смело беритесь за новые задачи — мы все делаем хорошо то, что делаем постоянно, и неохотно берёмся за что-то незнакомое, чего ещё ни разу не делали. Причина одна — мы боимся неудачи. Но неудачи — это часть обучения и развития
- Занимайтесь творчеством — оно развивает гибкость мышления



Проактивность

- ▶ Проактивность — это полезный навык для управления своей жизнью и карьерой, чтобы не плыть по течению, а самостоятельно задавать нужный путь и траекторию
- ▶ Если вы хотите выделяться из массы, быть лучшим, нужно быть проактивным — значит, делать больше, чем от вас ожидают

Как сформировать навык:

- Во-первых, забудьте фразу «это не моя работа». Есть вещи, которые напрямую не относятся к вашим обязанностям, но иногда возникают ситуации, когда ваша помощь необходима — в Agile люди помогают друг другу, берут на себя смежные функции: такой подход помогает команде быстрее и эффективнее выполнять задачи, а вам — расти
- Прежде чем идти к более старшим товарищам с вопросом — постарайтесь самостоятельно найти решение проблемы. В команде очень ценится подход, когда человек подходит не с «голым» вопросом, а уже владеет какой-то информацией, вариантами решения и спрашивает не «как это сделать», а «как сделать лучше и правильнее»



Навыки презентации

- ▶ Это очень нужно для то, чтобы демонстрировать результаты своей работы и делиться опытом с коллегами, а также помогает обучать других
- ▶ Полезно записывать свое выступление на камеру, чтобы посмотреть на себя со стороны. Многие начинающие спикеры совершают много непроизвольных движений, что мешает им и слушателям. Именно видео поможет вам заметить такие нюансы и попытаться избавиться от них

Как сформировать навык:

- Смотрите, как это делают другие. Просматривайте хорошие публичные выступления, запоминайте то, что вам показалось интересным и зацепило. Если есть возможность где-то выступить публично — делайте это, не стесняйтесь
- Тяните руку на лекциях и конференциях, выходите к доске, в круг, из ряда. Не прячьтесь за спинами других людей!



Менторинг

- ▶ Менторинг необходим, чтобы обмениваться знаниями с коллегами и помогать адаптироваться новичкам
- ▶ Помимо этого, обучение других укрепляет собственные знания

“В любой деятельности наступает момент, когда для того, чтобы вырасти как специалист, уже недостаточно развивать только свои собственные знания и навыки — необходимо передать их другим. Процесс обучения позволяет глубже заглянуть в самую суть своей работы, вычленить самое важное и систематизировать это до такого уровня, чтобы можно было понятно объяснить это другим”

Как сформировать навык:

Если вы умеете что-то, чего не умеют другие — пробуйте себя в роли репетитора. Это тренирует терпение и другие необходимые навыки



Вывод

- ▶ Возможно, для успеха в карьере нам достаточно только soft skills?
Конечно нет. В первую очередь, любые ИТ-компании ищут талантливых профессионалов, проверяют и оценивают ваши навыки специалиста (hard skills)
- ▶ Но только когда soft и hard skills гармонично сочетаются в одном человеке, он вырастает выдающимся профессионалом





Спасибо за внимание!



Экономика программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана
Каф. ИУ-7

baryshnikovam@mail.ru

Лекция 8

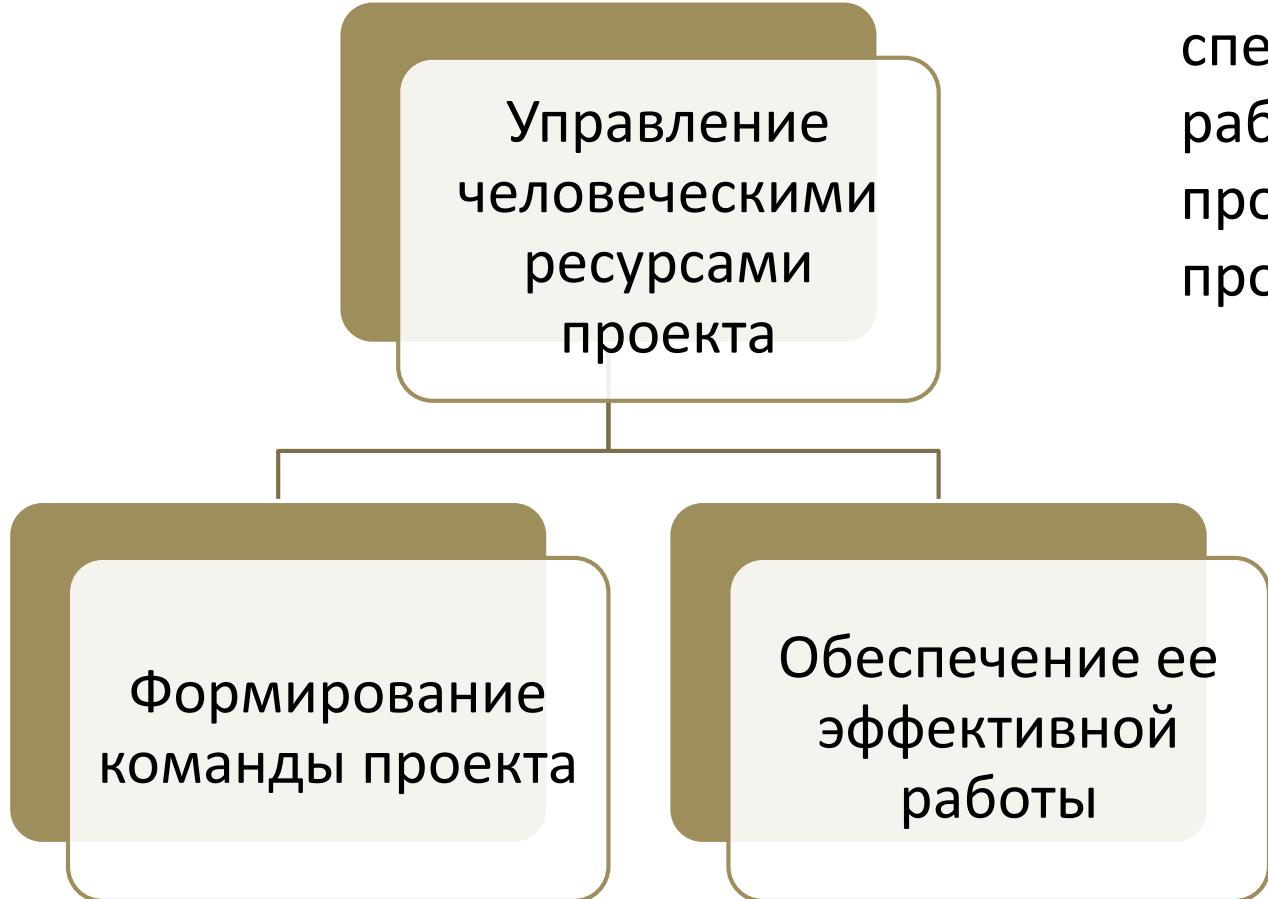
Управление человеческими ресурсами.
Формирование команды программного проекта.
Полномочия и ответственность. Роли для людей

Проект достигает запланированных
результатов только в том случае, если
эффективное управление в нем сочетается
с не менее эффективным руководством



Типовой состав участников проекта

- ▶ Инициатор проекта – физическое или юридическое лицо (группа лиц), являющееся автором главной идеи проекта и его предварительного обоснования. В качестве инициатора может выступать любой из участников проекта, но деловая инициатива по осуществлению проекта исходит либо от заказчика, либо от инвестора
 - ▶ Заказчик – физическое или юридическое лицо, являющееся владельцем и пользователем будущих результатов проекта
 - ▶ Инвестор - физическое или юридическое лицо (группа лиц), предоставляющее в любой форме финансовые ресурсы для проекта
 - ▶ Куратор проекта – представитель исполнителя, уполномоченный принимать решения о выделении ресурсов и внесении необходимых изменений в проект
 - ▶ Руководитель проекта (проект-менеджер) – физическое лицо, которому делегированы полномочия по руководству работами по реализации проекта (может быть представителем заказчика, инвестора или исполнителя)
 - ▶ Команда проекта – группа специалистов, формируемая в зависимости от потребностей, условий проектирования и организационной структуры выполнения проекта
-



Команда проекта – это группа специалистов, непосредственно работающих над осуществлением проекта и подчиненных руководителю проекта

Команда управления проектом – это часть команды проекта, которая отвечает за выполнение операций по управлению проектом

Действия по созданию команды проекта и организации ее эффективной работы

- ▶ определить потребности, численный и квалификационный состав персонала на весь период осуществления проекта
- ▶ произвести поиск и отбор кандидатур, оформить их прием на работу и увольнение
- ▶ осуществить планирование и распределение работников по рабочим местам
- ▶ организовать обучение и повышение квалификации
- ▶ распределить роли и ответственности в команде
- ▶ создать условия и рабочую атмосферу для коллективной работы
- ▶ предупреждать и разрешать возникающие конфликты
- ▶ разработать механизмы мотивации и оплаты



Факторы, которые следует учитывать при формировании команды программного проекта

- ▶ *Специфика проекта* – определяет формальную структуру команды, которая утверждается руководством: ролевой состав; перечень знаний, умений и навыков, которыми должны владеть члены команды; сроки, этапы, виды работ по проекту
- ▶ *Организационно-культурная среда* – определяется способами организации и протекания процессов командного взаимодействия (такими как координация, коммуникация, деятельность по разрешению конфликтов и принятию решений, налаживание внешних связей)
- ▶ *Особенности личного стиля взаимодействия* руководителя команды с другими ее членами – менеджер проекта должен быть гибким и уверенным в себе и в своих сотрудниках. Его влияние в команде должно основываться не на статусе или положении, а на профессионализме и компетенции



Особенности управления командой разработчиков ПО

- ▶ *узкая специализация сотрудников.* Менеджер проекта подбирает команду, в которой каждый из участников максимально эффективен в конкретной области. При этом никто из них не может видеть проблему в целом, что существенно усложняет задачи по оптимизации и интеграции системы
- ▶ *творческий характер труда программиста.* Разработка ПО — творческий процесс, разработчики являются креативными личностями, которые способны привносить в общее дело энтузиазм, инициативу и собственные нетривиальные решения. При наличии сильной мотивации и ясной цели они, как правило, готовы работать с полной самоотдачей. Это значит, что управление персоналом в программных проектах следует организовывать по целям, а не по заданиям
- ▶ *высокая мобильность сотрудников.* В современных условиях спрос на квалифицированных программистов существенно превышает предложение. Руководитель должен быть готов к внезапному уходу из команды (фирмы) любого из сотрудников. Процесс разработки следует организовать так, чтобы это не вызвало катастрофических последствий для проекта. При этом необходимо учесть два аспекта: возможность утраты необходимого работника и возможность потери программного кода. Программисты часто не понимают, что программные продукты, разработанные в рамках проекта организации, им не принадлежат

Особенности управления командой разработчиков ПО

- ▶ *постоянное повышение квалификации*, поскольку технологии разработки ПО весьма быстро морально устаревают, а качественно новые версии инструментальных средств появляются достаточно часто, иногда с периодичностью раз в полгода
- ▶ *высокая самооценка программиста*. Узкая специализация и высокая профессиональная квалификация в конкретной области часто вызывают у сотрудников завышенную самооценку своих возможностей, что требует от руководителя:
 - постоянного изучения реальных возможностей своих сотрудников и корректировку их самооценки;
 - способности убеждать и мотивировать членов команды, поскольку сотруднику с высокой самооценкой трудно что-либо приказать, его необходимо убедить, что бывает непросто, в силу того что сам руководитель вряд ли может быть авторитетом в области, где сотрудник является узким специалистом

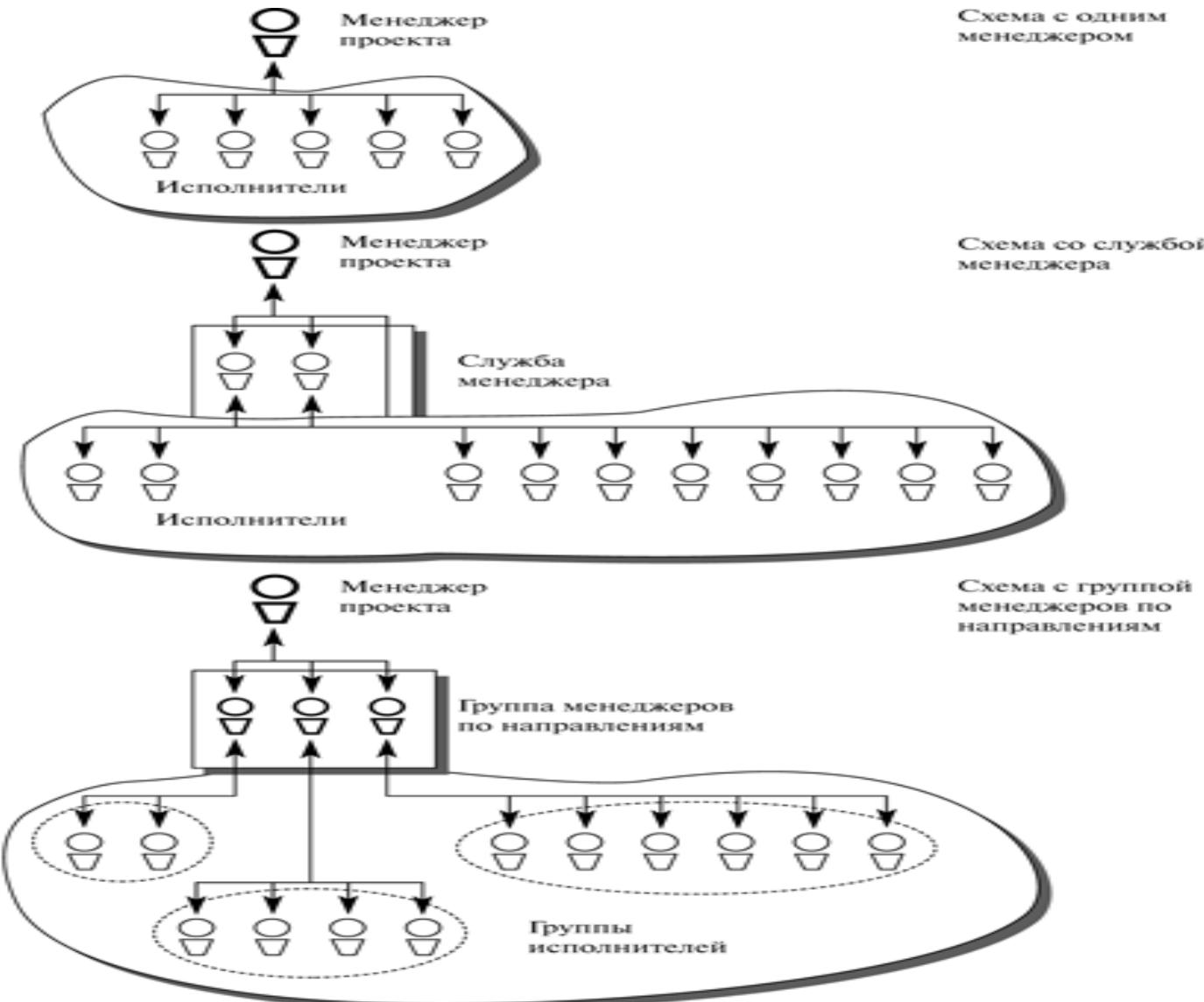


Адаптивные организационные структуры

Организационная структура оказывает большое влияние на принципы формирования команды проекта, а также на взаимоотношения участников проекта внутри команды. Для достижения успеха программного проекта часто необходимо иметь возможность оперативно реагировать на изменения внешней среды, внедрять новые технологии, корректировать требования к конечному результату. Этим требованиям в наибольшей степени отвечают так называемые адаптивные организационные структуры

- ▶ Проектная организация - это временная структура, создаваемая для решения конкретной задачи. Ее цель состоит в том, чтобы собрать в одну команду самых квалифицированных сотрудников организации для осуществления конкретного проекта в установленные сроки с заданным уровнем качества, не выходя за пределы установленной сметы. Когда проект завершен, команда распускается
- ▶ Матричная структура получается при наложении проектной структуры на постоянную для данной организации функциональную структуру. В матричной организации члены проектной группы подчиняются как руководителю проекта, так и руководителю того функционального подразделения, в котором они работают постоянно

Возможные схемы организации менеджмента проекта

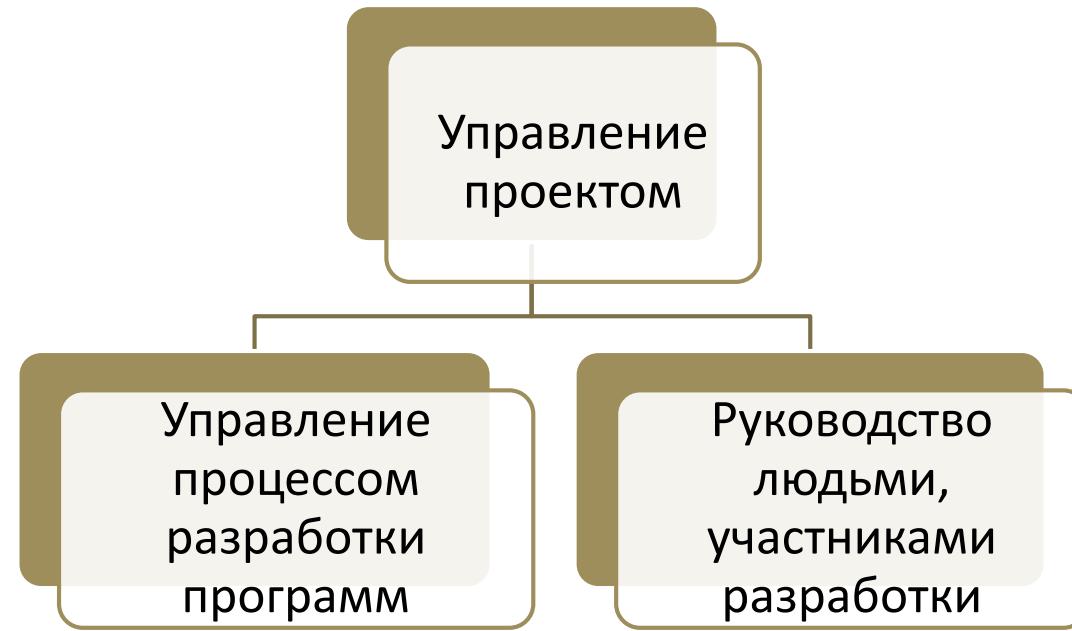


Используется в небольших некритичных проектах. Данная схема позволяет исключить проблемы согласования противоречий внутри команды, обеспечивает централизованную ответственность за проект перед заказчиком

Применяется в более масштабных проектах, что позволяет освободить менеджера проекта от рутинного контроля выполнения отдельных задач. При этом по своему статусу помощники менеджера являются обычными работниками

В более сложных проектах образуется специальная группа менеджеров, ответственных за разные разграниченные сферы проекта, члены которой получают соответствующие полномочия в своих сферах ответственности (обозначены пунктирными овалами)

Организационная схема управления проектом



Разработка программного обеспечения в большинстве случаев может рассматриваться как коллективный труд специалистов, направленный на удовлетворение потребности пользователей в автоматизации их деятельности, поэтому, как и любой другой коллективный труд, она требует координации и управления



«Три кита» управления



Полномочия представляют собой ограниченное право использовать ресурсы организации и направлять усилия некоторых ее сотрудников на выполнение определенных задач. Полномочия даются должности (или роли), а не индивиду, который занимает (выполняет) ее в данный момент

- ▶ *Ответственность* - обязательство выполнять имеющиеся задачи и отвечать за их удовлетворительное разрешение
- ▶ *Власть* – формальное право или реальная возможность требовать от кого-либо следовать определенному курсу действий. Именно ответственность служит ограничению и мотивации власти

Полномочия определяют, что лицо, занимающее какую-то должность, **имеет право** делать, власть же определяет, что оно **действительно может** делать.



Делегирование полномочий

- ▶ Делегирование - это передача задач и полномочий лицу, которое принимает на себя ответственность за их выполнение
- ▶ Умение делегировать часть своих полномочий превращает человека в руководителя
- ▶ Делегируются только полномочия, ответственность не может быть делегирована
- ▶ Руководитель любого уровня только тогда будет иметь возможность повлиять на деятельность людей, от которых зависит выполнение задачи, когда он будет располагать требующимися для этого ресурсами

Пределы полномочий внутри организации определяются:

- 1) политикой
- 2) процедурами
- 3) правилами
- 4) должностными инструкциями, изложенными письменно или в устной форме

Историческая справка: став президентом Соединенных штатов Г. Трумэн повесил в своем кабинете плакат, который гласил: «Больше ответственность сваливать не на кого»



Процессы управления человеческими ресурсами

- ▶ планирование человеческих ресурсов – определение и документальное оформление ролей, ответственности и подотчетности, а также создание плана обеспечения проекта персоналом
- ▶ набор команды проекта – привлечение человеческих ресурсов, необходимых для выполнения проекта
- ▶ развитие команды проекта – повышение квалификации членов команды проекта и укрепление взаимодействия между ними с целью повышения эффективности исполнения проекта
- ▶ управление командой проекта – контроль за эффективностью членов команды проекта, обеспечение обратной связи, разрешение проблем и координация изменений, направленных на повышение эффективности проекта



Факторы, учитываемые при планировании человеческих ресурсов

- ▶ *Организационные.* Какие организации или структурные подразделения привлекаются к участию в проекте? Какие механизмы взаимодействия существуют на данный момент между ними? Каковы сложившиеся на данный момент формальные и неформальные отношения между ними?
 - ▶ *Технические.* Какие различные навыки и специальности необходимы для выполнения данного проекта? Существует ли необходимость в обеспечении координации между языками программирования, инструментальными средами разработки программ или различными типами оборудования? Существуют ли какие-либо специфические сложности при переходе от одной фазы жизненного цикла к другой?
 - ▶ *Межличностные.* Какие официальные и неофициальные отношения подотчетности существуют на данный момент между кандидатами в члены команды проекта? Какие культурные или языковые отличия между членами команды могут оказать влияние на рабочие взаимоотношения? Каков существующий на данный момент уровень доверия и уважения между потенциальными членами команды?
 - ▶ *Логистические.* Какое расстояние отделяет возможных кандидатов в члены команды проекта друг от друга? Находятся ли эти люди в различных зданиях, часовых поясах или странах?
 - ▶ *Политические.* Каковы цели и интересы каждого из потенциальных участников проекта? Какие люди или группы людей имеют неформальное влияние в областях, представляющих важность для проекта? Какие существуют неформальные связи между потенциальными участниками проекта?
-

Форматы определения ролей и ответственности



Иерархическая организационная диаграмма

Матричная диаграмма ответственности

Роль	<hr/> <hr/>
Ответственность	<hr/> <hr/>
Полномочия	<hr/> <hr/>

Текстовый формат

Пример диаграммы RACI (Responsible, Accountable, Consult and Inform - Ответственный, Подотчетный, Проконсультированный и Информированный)

Операция	СОТРУДНИКИ				
	Иванов	Петров	Морозова	Костин	Серегина
Определение требований	П	О	И	И	И
Проектирование	И	П	О	К	К
Разработка	И	П	О	К	К
Тестирование	П	И	И	О	И

О – ответственный
П – подотчетный
К – проконсультированный
И – проинформированный



Результаты процесса планирования человеческих ресурсов

- ▶ Организационная диаграмма проекта
- ▶ План обеспечения проекта персоналом
 - Набор персонала
 - Расписание
 - Критерии освобождения ресурсов
 - Обучение персонала
 - Поощрение и премирование

Планирование человеческих ресурсов проводится с целью определения ролей, ответственности и подотчетности в проекте, а одним из основных его результатов является разработка плана обеспечения проекта персоналом



Результаты процесса планирования человеческих ресурсов

- ▶ Организационная диаграмма проекта
- ▶ План обеспечения проекта персоналом:
 - набор персонала (будут ли для реализации проекта задействованы имеющиеся человеческие ресурсы организации или они будут набираться извне на контрактной основе? Должны ли члены команды работать в одном месте или они могут работать удаленно? Какова стоимость, соответствующая каждому уровню квалификации специалистов, необходимых для проекта? Насколько отдел кадров организации может помочь команде управления проектом?)
 - расписание (указываются временные рамки задействования членов команды проекта, индивидуально или по группам, а также указывается время начала операций по набору персонала)
 - критерии освобождения ресурсов (когда члены команды освобождаются от участия в проекте согласно выверенному расписанию, то при этом исключаются выплаты сотрудникам, уже выполнившим свою долю работы в проекте, и таким образом снижаются затраты на проект)
 - обучение персонала (разрабатывается, если квалификация членов команды, привлекаемых для участия в проекте, является недостаточной для выполнения возлагаемых на них ролей)
 - поощрение и премирование (ясные критерии премирования и спланированная система премий помогут стимулировать и поддержать желаемую производительность людей, занятых в проекте. Чтобы поощрение и премирование было эффективным, оно должно основываться на операциях и производительности, которые находятся в сфере ответственности данного лица)



Набор команды проекта

Набор команды проекта – это процесс привлечения человеческих ресурсов, необходимых для выполнения проекта. Набор происходит из всех доступных источников, как внутренних, так и внешних. При наборе персонала для участия в проекте кадровые службы ориентируются либо на схему распределения ролей и ответственности, которая содержит позиции, навыки и квалификацию специалистов, которые требуются для проекта, либо на организационные диаграммы проекта

Назначение персонала требует согласования с:

- ▶ функциональными руководителями – чтобы гарантировать, что проект будет обеспечен соответствующим штатом квалифицированных сотрудников на требуемый период времени и чтобы члены команды проекта могли работать на проекте до полного окончания возложенных на них работ
- ▶ другими командами управления проектом в рамках исполняющей организации – чтобы обеспечить проект дефицитными ресурсами или узко-профильными специалистами

Если у организации для выполнения проекта не хватает штатных специалистов, то требуемые услуги можно получить за счет найма консультантов или передачи работ сторонним организациям на условиях субподряда



Виртуальные команды

Дают возможность:

- ▶ формировать команды из числа сотрудников одной компании, проживающих в различных регионах
- ▶ добавлять в состав команды специалистов, даже если они находятся в другом регионе
- ▶ привлекать к участию в проекте сотрудников, работающих дома
- ▶ формировать команды из исполнителей, работающих в разные смены или в разные часы
- ▶ привлекать к участию в проекте инвалидов
- ▶ браться за выполнение проектов, реализация которых в иных условиях была бы невозможна из-за высоких командировочных расходов



Развитие команды проекта

Целями развития команды проекты являются:

- ▶ развитие навыков членов команды в части выполнения задач и/или операций проекта
- ▶ укрепление чувства доверия и сплоченности среди членов команды для повышения продуктивности ее работы

Развитие команды проекта предусматривает повышение квалификации членов команды проекта и укрепление взаимодействия между ними. Кроме этого важны навыки межличностных отношений и наличие ясных и четких правил поведения, которые принимаются всеми членами коллектива

Обучение членов команды может носить как официальный, так и неофициальный характер и осуществляться в форме направления на курсы повышения квалификации или стажировки, дистанционного обучения в режиме онлайн, обучения на рабочем месте под руководством другого члена команды проекта, наставничества и тренингов



Управление командой проекта

- ▶ контроль за деятельностью членов команды проекта
- ▶ обеспечение обратной связи
- ▶ разрешение проблем
- ▶ координация изменений, направленных на повышение эффективности реализации проекта

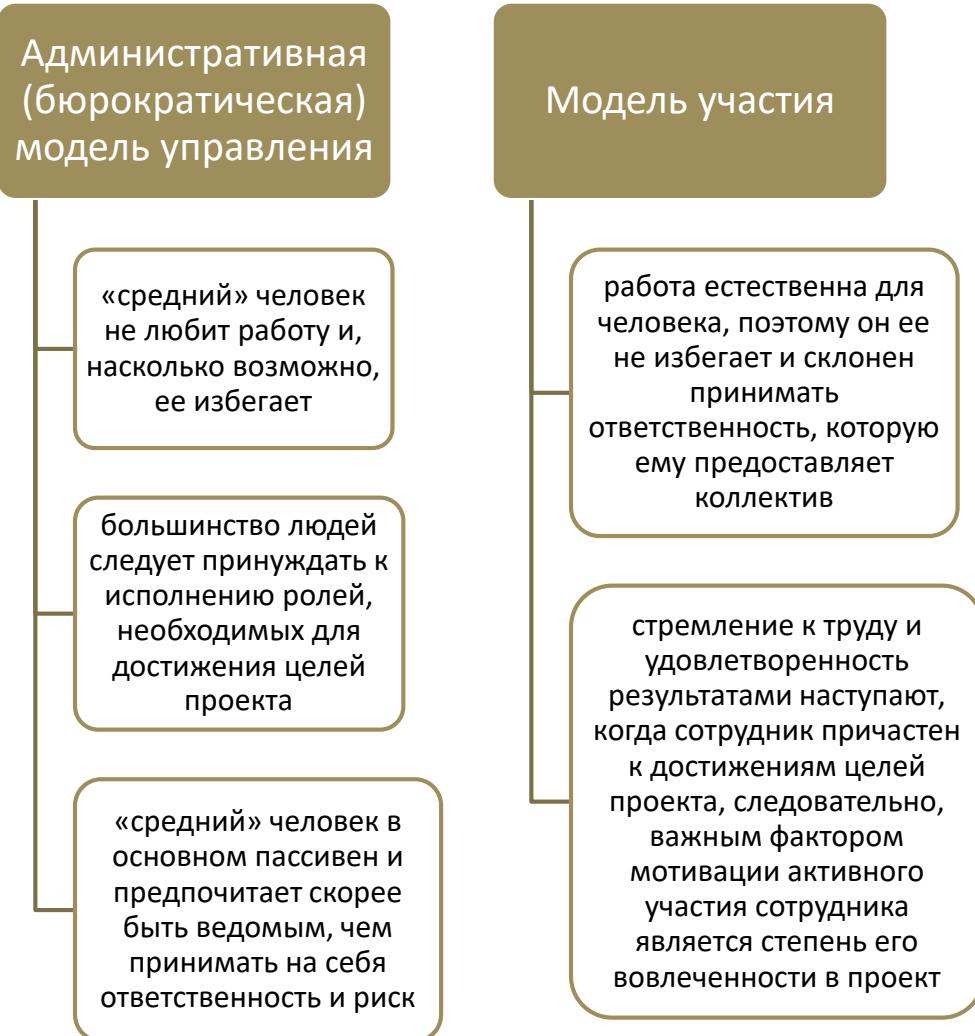
Для оценки эффективности работы команды могут использоваться следующие показатели:

- ▶ повышение навыков членов команды, что позволяет им более эффективно выполнять порученные операции
- ▶ укрепление сплоченности (team-building), что положительно сказывается на работе всей группы
- ▶ сокращение текучести кадров

Для того чтобы быть в курсе процесса выполнения работ и настроений, царящих среди членов команды проекта, руководство проектом должно чаще использовать такие методы как наблюдение и обсуждение. К корректирующим действиям по управлению человеческими ресурсами относятся кадровые перестановки, проведение дополнительных тренингов и меры дисциплинарного воздействия



Основные модели управления командой проекта и их предпосылки



На выбор модели управления влияют:

- 1) *используемые производственные технологии.* При разработке ПП под заказ, где работа носит в основном творческий характер, целесообразно использовать модель участия. При тиражном производстве программных продуктов, где бизнес-процессы жестко заданы и заранее определены, более эффективной будет бюрократическая модель
- 2) *характер поведения внешней среды.* Если потребности пользователей неизменны и рынок предсказуем, то лучше использовать бюрократическую модель организации управления. Быстро изменяющиеся потребности внешней среды требуют гибкого управления, что соответствует модели участия
- 3) *размер организации.* Чем крупнее фирма, тем сильнее проявляется тенденция к использованию формальной бюрократической модели управления

Характеристики бюрократической модели

- ▶ организация проектной деятельности по достижению целей фирмы в виде зафиксированных формальных ролей и обязанностей сотрудников и четкое распределение ответственности
- ▶ властная пирамида управления проектом, построенная по принципу иерархии
- ▶ выполнение процессов и работ по проекту в соответствии с утвержденным регламентом
- ▶ строго формализованное управление проектом со стороны менеджеров, исключающее эмоциональные факторы при принятии решений

Достоинства	Недостатки
четкая организация труда	игнорирование активной роли сотрудников при выполнении проекта
разграничение ролей, обязанностей и ответственности	отсутствие мотивации к творческому труду
хорошо организованная система контроля	невосприимчивость к изменениям (появлению новых типов проектов и технологий, необходимости оперативного реагирования на изменение требований пользователей и рынка и т.д.)



Характеристики модели участия

- ▶ независимые модули разрабатываются отдельными членами команды, архитектура программного продукта (и другие системные решения) – результат коллективного обсуждения
- ▶ принятное коллективное решение является обязательным для всех (даже если кто-то из членов команды с ним не согласен)
- ▶ менеджер проекта обязательно участвует в процессе коллегиальной выработки решений
- ▶ осуществляется контроль конструктивности обсуждений и обеспечивается возможность активного участия в них для всех сотрудников
- ▶ существует распределенная ответственность за качество своих разработок и коллективная — за принятие коллегиальных решений (отвечают все, кто обсуждал, вырабатывал, принимал)
- ▶ явно отсутствует специализация — сотрудники могут при необходимости заменить друг друга

Достоинства	Недостатки
модель эффективна в случае, когда для решения проблемы требуется поиск новых подходов, методов, идей и средств	сотрудники, способные к генерации идей, редко обладают терпением в доведении идей до полной реализации, а творческая активность переходит в конкуренцию — сначала идей, а потом личностей



Как влияют на выбор модели управления персональные и корпоративные стратегии компаний

На выбор модели управления сильно влияют, в том числе, положение компании на рынке и управленческий «темперамент» ее сотрудников:

- ▶ Если компания хочет путем агрессивного управления добиться наибольшего успеха на рынке, она будет стремиться создать нестабильность на рынке и захватить себе большую его часть. Очевидно, в этом случае целесообразнее всего использовать модель участия
- ▶ Если компания – лидер на рынке, то она испытывает большое давление со стороны конкурентов, поэтому стремится к относительной стабильности. В этом случае управление должно быть в большей мере ориентировано на сохранение статус-кво, что может быть обеспечено только четкой регламентацией действий всех сотрудников и организации в целом, следовательно лучше сработает бюрократическая модель



Выбор руководителя проекта

Критерии отбора:

- | | |
|--|--|
| 1) образование и опыт | 2) лидерство и стратегическое мышление |
| 3) техническая компетентность | 4) умение работать с людьми |
| 5) доказанные способности к управлению | |

При отборе потенциального руководителя проекта учитываются:

- ▶ знания используемых в проекте технологий
- ▶ знания технических инструментов и методов
- ▶ знания соответствующих рынков, заказчиков и требований
- ▶ знания о применениях программных продуктов
- ▶ знания тенденций развития технологии
- ▶ знакомство со смежными техническими областями
- ▶ знакомство с людьми, составляющими часть «профессионального сообщества»



Умение работать с людьми

Руководитель проекта должен уметь:

- ▶ заинтересовывать, внушать, ободрять и обучать
- ▶ внимательно выслушивать предложения и поддерживать «обратную связь»
- ▶ настойчиво - но не агрессивно! - соотносить с другими и выделять среди других потребности, заботы и межличностные конфликты, связанные с проектом, уметь их предотвращать и разрешать
- ▶ сообщать жесткие решения, даже если они задевают интересы других
- ▶ проявлять гибкость - правильно вести себя в различных ситуациях (уметь хорошо играть разные роли)



Подбор команды проекта – критерии отбора

- ▶ компетентность в предметной области, определяющей специфику проекта
 - ▶ способность делегировать полномочия и разделять ответственность
 - ▶ ориентированность на выполнение задачи
 - ▶ способность переходить от одного вида работы к другому в зависимости от графика работ и производственной необходимости
 - ▶ готовность признавать ошибки и воспринимать замечания
 - ▶ способность к пониманию планов, готовность работать в условиях жесткого графика и лимита ресурсов
 - ▶ готовность к сверхурочной работе, если необходимо в интересах проекта
 - ▶ способность доверять, помогать другим и принимать помощь
 - ▶ умение быть игроком в команде, а не героем-одиночкой
 - ▶ предпримчивость, но при этом восприятие советов и предложений
 - ▶ способность работать с двумя и более начальниками
 - ▶ способность работать без и вне формальных иерархий и систем полномочий
 - ▶ знания и опыт в области управления проектами
-

Подходы к формированию команд программных проектов

- ▶ По результатам множества опросов менеджеров проектов в России и за рубежом, до 80% успеха при реализации проектов обусловлено слаженной работой проектной команды, которая, в свою очередь, обеспечивается верным распределением ролей среди участников
 - ▶ Существенный вклад в успех вносят также такие факторы как численность команды и квалификация ее членов (руководители программных проектов давно уже поняли, как значительны различия в производительности хороших и плохих программистов)
 - ▶ Как только программирование перестает быть уделом одиночек, а превращается в коллективный труд, требующий обмена данными и скоординированной работы отдельных частей программы, механическое увеличение численности команды не только не приводит к сокращению сроков разработки, а зачастую растягивает проект во времени и приводит к дополнительным затратам на обеспечение коммуникации и на устранение последствий плохой организации этой коммуникации
 - ▶ Руководители программных проектов стремятся максимально ограничить число людей, работающих над системой, так как при таком подходе проще обеспечить концептуальное единство проекта и выработать согласованную позицию участников относительно стратегии его реализации
 - ▶ Концепция маленькой энергичной группы имеет один существенный недостаток – это *слишком медленно для действительно больших систем* (есть риск, что растянув процесс разработки программной системы на годы, вы создадите продукт, потребность в котором давно уже отпала)
-



Команда программного проекта по Харлану Миллзу

- ▶ *Хирург* или главный программист. Определяет функциональные спецификации и показатели производительности программы, проектирует ее, пишет код и отлаживает его, готовит документацию. Должен обладать хорошими профессиональными навыками и иметь большой опыт практической работы
- ▶ *Второй пилот* – правая рука хирурга. Способен выполнить любую часть работы, но не столь опытен. Представляет свою бригаду на дискуссиях, взаимодействует с другими бригадами. До тонкостей знает всю программу, ищет альтернативные стратегии проектирования. Он может даже программировать, но не несет ответственности ни за одну часть программы
- ▶ *Администратор* – занимается деньгами, людьми, компьютерами, входит в контакты с администрацией всей организации. Один администратор может обслуживать две бригады
- ▶ *Редактор* – получает рукопись хирурга и оценивает (может быть даже критикует) ее, перерабатывает, снабжает ссылками и библиографией, готовит различные версии документов и наблюдает за их размножением и распространением



Команда программного проекта по Харлану Миллзу

- ▶ *Два секретаря* – у администратора и редактора
- ▶ *Архивариус или делопроизводитель* – отвечает за ведение всей технической документации в бригаде, осуществляет контроль всех изменений, вносимых как в программный код, так и в документацию
- ▶ *Инструментальщик*. Команде, занимающейся созданием ПО, помимо стандартных сред разработки может потребоваться дополнительный инструментарий, набор вспомогательных средств, позволяющих автоматизировать рутинные операции и повысить эффективность разработки. Их приобретение, адаптация и усовершенствование составят круг обязанностей опытного системного программиста - инструментальщика
- ▶ *Контролер или отладчик* – готовит набор подходящих тестов для отладки частей программы по мере ее написания и для отладки программы как единого целого
- ▶ *Языковед* – специалист, хорошо владеющий выбранным для написания программы языком программирования



Преимущества организации команды проекта по принципу хирургической бригады

Группа из 10 человек	Хирургическая бригада Х. Милза
Вся работа разделена между сотрудниками, и каждый из них отвечает за разработку и реализацию своей части	В хирургической бригаде и хирург, и второй пилот в курсе всего проекта и всей программы
Требуются специальные организационные усилия по выстраиванию коммуникации и взаимодействия членов группы	Минимум усилий на организацию коммуникации, координацию и взаимодействие
Все сотрудники равны, поэтому неизбежные различия в оценках требуют постоянных обсуждений и компромиссов	Нет различий в интересах, а противоречия во мнениях разрешаются самим хирургом единолично
Поскольку работа и ресурсы разделены, различия в суждениях, хотя и подчинены общей стратегии, усугубляются противоположностью интересов	Строгое распределение ролей между сотрудниками бригады является ключом к повышению ее производительности



Подход Центра объектно-ориентированной технологии компании IBM

- ▶ **Заказчик (Customer)** — реально существующий (в организации, которой подчинена команда, или вне ее) инициатор разработки или кто-либо иной, уполномоченный принимать результаты (как текущие, так и окончательные) разработки
 - ▶ **Планировщик ресурсов (Planner)** — выдвигает и координирует требования к проектам в организации, осуществляющей данную разработку, а также развивает и направляет план выполнения проекта с точки зрения организации
 - ▶ **Менеджер проекта (Project Manager)** — отвечает за развитие проекта в целом, несет ответственность за распределение заданий и ресурсов, за соответствие результатов установленным требованиям
 - ▶ **Руководитель команды (Team Leader)** — производит техническое руководство командой в процессе выполнения проекта. Для больших проектов возможно привлечение нескольких руководителей подкоманд, отвечающих за решение частных задач
 - ▶ **Архитектор (Architect)** — отвечает за проектирование архитектуры системы, согласовывает развитие работ, связанных с проектом
 - ▶ **Проектировщик подсистемы (Designer)** — отвечает за проектирование подсистемы или категории классов, определяет реализацию и интерфейсы с другими подсистемами
 - ▶ **Эксперт предметной области (Domain Expert)** — изучает сферу приложения, поддерживает направленность проекта на решение задач данной области
-

Подход Центра объектно-ориентированной технологии компании IBM

- ▶ *Разработчик (Developer)* — реализует проектируемые компоненты, владеет и создает специфичные классы и методы, осуществляет кодирование и автономное тестирование, строит продукт
- ▶ *Разработчик информационной поддержки (Information Developer)* — создает документацию, сопровождающую продукт, когда выпускается версия. Включаемые в нее инсталляционные материалы, равно как ссылочные и учебные, а также материалы помощи представляются на бумажных и электронных носителях
- ▶ *Специалист по пользовательскому интерфейсу (Human Factors Engineer)* — отвечает за удобство применения системы. Работает с заказчиком, чтобы удостовериться, что пользовательский интерфейс удовлетворяет требованиям
- ▶ *Тестировщик (Tester)* — проверяет функциональность, качество и эффективность продукта. Строит и исполняет тесты для каждой фазы развития проекта
- ▶ *Библиотекарь (Librarian)* — отвечает за создание и ведение общей библиотеки проекта, которая содержит все проектные рабочие продукты, а также за соответствие рабочих продуктов стандартам

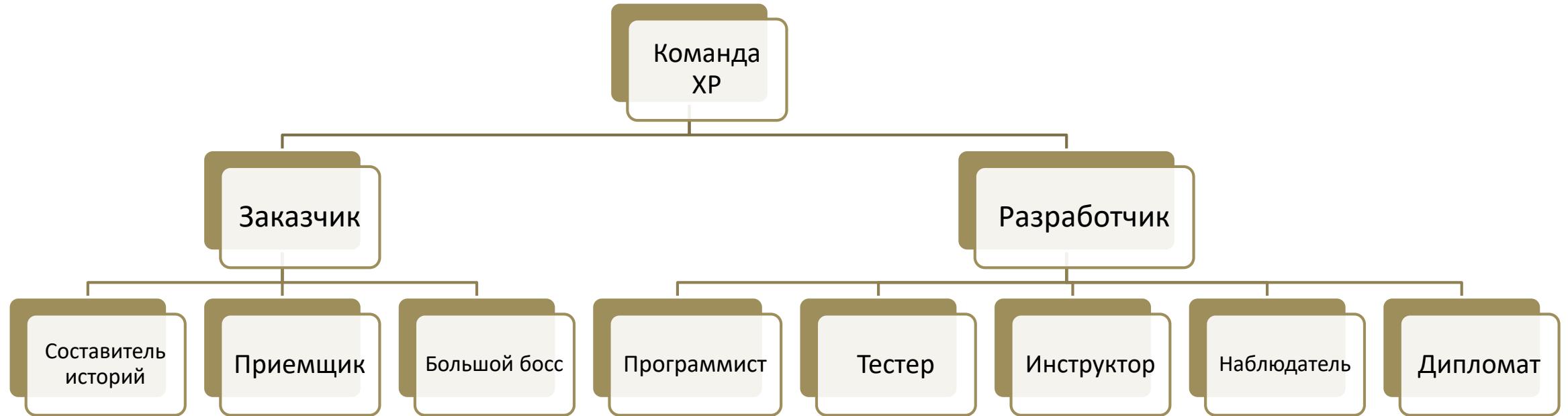


Возможность совмещения ролей

- ▶ В большинстве случаев заказчик и планировщик ресурсов являются внешними по отношению к проекту участниками разработки, поэтому их роли практически никогда не совмещаются с другими
- ▶ Менеджер проекта по своему назначению является выделенным лицом команды. Он осуществляет взаимодействие с заказчиком и планировщиком ресурсов и распределяет работы среди членов команды. Кроме того, он обладает полной информацией о декомпозиции проекта, поэтому совмещение его роли с ролью архитектора проекта является желательным и довольно частым
- ▶ Возможно совмещение ролей руководителя команды и архитектора, особенно если команда имеет опыт предшествующей работы со своим руководителем, но при условии не слишком высокой сложности задач декомпозиции для данного проекта
- ▶ Совмещение ролей руководителя команды и менеджера допустимо, но лишь тогда, когда осознается и учитывается противоречивость целевых установок этих ролей: руководитель команды действует в условиях, которые формируются менеджером
- ▶ Крайне нежелательно совмещение ролей руководителя команды и проектировщика какой-либо подсистемы, так как в этом случае у руководителя могут сложиться предпочтения в пользу «своего» компонента
- ▶ Совмещение ролей различных разработчиков — обычное дело для больших проектов, так как является частью распределения работ по исполнителям
- ▶ Желательно, чтобы роли тестировщиков поручались независимым специалистам в предметной области, а не членам команды, однако приемлемо так называемое «перекрестное» совмещение ролей разработчиков и тестировщиков, когда разработчики независимых компонентов проекта testируют функциональность друг друга



Команда XP проекта – роли для людей



Команда XP проекта – роли для людей

Заказчик имеет следующие права и обязанности:

- ▶ зафиксировать сроки выпуска версий продукта
- ▶ принимать решения относительно запланированных составляющих программы
- ▶ знать ориентировочную стоимость каждой функциональной составляющей
- ▶ принимать важные бизнес-решения
- ▶ знать текущее состояние системы
- ▶ изменять требования к системе, когда это действительно важно

Разработчик наделён следующими правами и обязанностями:

- ▶ получить достаточное знание вопросов, которые должны быть запрограммированы
- ▶ иметь возможность выяснения деталей в процессе разработки
- ▶ предоставлять ориентировочные, но откровенные оценки трудозатрат на каждую функциональную часть или историю пользователя
- ▶ корректировать оценки в пользу более точных в процессе разработки
- ▶ предоставлять оценку рисков, связанных с использованием конкретных технологий



Команда XP проекта – роли для людей

Страна заказчика

- ▶ *Составитель историй* - специалист предметной области, обладающий способностями доступно изложить и описать требования к разрабатываемой системе. Этот человек или группа людей ответственны за написание историй пользователя и прояснения недопонимания со стороны программистов
- ▶ *Приёмщик* - человек, контролирующий правильность функционирования системы. Хорошо владеет предметной областью. В обязанности входит написание приёмочных тестов
- ▶ *Большой босс* - следит за работой всех звеньев, от разработчиков до конечных пользователей. Он контролирует внедрение системы и сопутствующие организационные моменты. Может быть также инвестором проекта



Команда XP проекта – роли для людей

Страна разработчика

- ▶ *Программист* - человек, занимающийся кодированием и проектированием на низком уровне. Он достаточно компетентен для решения текущих задач разработки и предоставления правдивых оценок запланированных задач. Но при этом он должен обладать умением работать в паре, привычкой к простоте, умением и желанием постоянно переделывать код и рассматривать систему как общую собственность, которая принадлежит всей команде
- ▶ *Тестер* – помогает заказчику в подборе и написании функциональных тестов, отвечает за регулярный их запуск и оповещает команду о результатах тестирования. Кроме того, он следит за правильностью работы тестирующих инструментов
- ▶ *Инструктор* – отвечает за весь процесс разработки. Это опытный разработчик, хорошо владеющий всем процессом разработки и его методиками. Несёт ответственность за обучение команды аспектам процесса разработки. Внедряет и контролирует правильность выполнения методик используемого процесса. Обращает внимание команды на важные, но по каким-то причинам упущенные моменты разработки
- ▶ *Наблюдатель* (ревизор) - член команды разработчиков, пользующийся доверием всей группы (совесть команды), который следит за прогрессом разработки. Он сравнивает предварительные оценки трудозатрат и реально затраченные усилия, выводя количественные показатели работы команды, такие как средняя скорость и процентное соотношение выполненных и запланированных задач. Наблюдатель выполняет функции историка команды, он ведет журнал результатов функционального тестирования и журнал обнаруженных дефектов. При этом он должен уметь собирать нужную ему информацию, не беспокоя весь остальной процесс больше, чем это необходимо
- ▶ *Дипломат* - коммуникабельная личность, инициирующая общение между членами команды. Дипломат регулирует и упрощает общение между заказчиками и разработчиками. Дипломат не может навязывать своего мнения дискутирующим сторонам



Команда XP проекта – роли для людей

Внешние роли

- ▶ *Консультант* - специалист, обладающий конкретными техническими навыками, для помощи разработчикам в трудно разрешимых задачах. Обычно привлекается со стороны. Он снабжается тестами, которые должны подсказать, когда проблему можно считать решенной. Однако при этом команда не дает консультанту просто уйти и решить проблему в одиночку. Ее задача – получить от консультанта все необходимые знания для того, чтобы в будущем решить проблему своими силами, поэтому некоторые члены команды постоянно будут сидеть рядом с консультантом, задавать множество вопросов и пытаться понять, нельзя ли сделать предлагаемое консультантом решение более простым



Функциональные ролевые группы в команде проекта в соответствии с моделью MSF

Группа управления проектом:

- управление процессом разработки с целью получения готового продукта в отведенные сроки
- регулирование взаимоотношений и коммуникаций внутри проектной группы
- контроль временного графика проекта и подготовка отчета о его состоянии
- организация управления рисками

Группа разработки программного продукта:

- разработка физического дизайна программной системы
- оценивание времени и ресурсов, необходимых для реализации каждого элемента дизайна
- разработка элементов
- подготовка продукта к внедрению
- консультирование команды по технологическим вопросам

Группа проектирования архитектуры:

- формулирование спецификации решения и разработка его архитектуры
- определение структуры развертывания (внедрения) решения

Группа тестирования:

- поиск и обнаружение дефектов
- разработка стратегии и планов тестирования
- тестирование



Функциональные ролевые группы в команде проекта в соответствии с моделью MSF

Группа управления выпуском:

- представление интересов отделов поставки и обслуживания продукта
- организация снабжения проектной группы
- организация внедрения продукта
- организация сопровождения и инфраструктуры поставки

Группа обеспечения связи с заказчиком:

- представление интересов потребителя в команде
- организация работы с требованиями пользователей
- нахождение компромиссов, относящихся к удобству использования и потребительским качествам продукта
- определение требований к системе помощи и ее содержанию
- разработка учебных материалов и обучение пользователей

Виды специализации сотрудников в соответствии с моделью MSF:

- ▶ Менеджер проекта
- ▶ Архитектор
- ▶ Бизнес-аналитик
- ▶ Разработчик
- ▶ Тестировщик
- ▶ Менеджер по работе с заказчиком

Не люди должны строиться под выбранную модель процесса, а модель процесса должна выбираться под возможности конкретной команды, чтобы обеспечить наивысшую эффективность ее работы



Основные положения мотивации сотрудников

Мотивация – методы воздействия на людей с целью получения желаемого результата



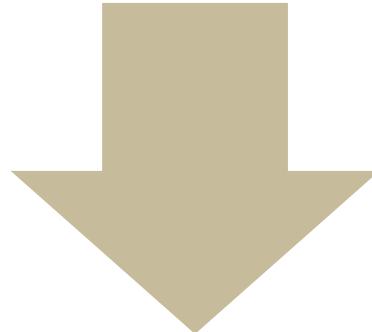
Потребности человека по А. Маслоу:

- Физиологические потребности
- Потребности в безопасности
- Социальные потребности
- Потребность в уважении
- Потребность в самовыражении

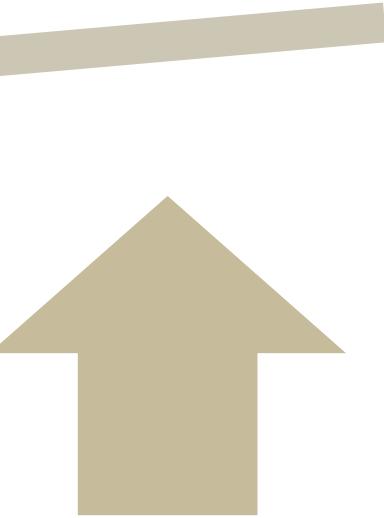
Побуждение - порождаемое потребностью ощущение недостатка чего-либо, имеющее определенную направленность

Вознаграждение – все, что человек считает ценным для себя

Факторы, определяющие мотивацию человека



Деньги, выгода, комфорт и тому подобное являются необходимыми, но недостаточными факторами для того, чтобы заставить людей полюбить свою работу и дать им необходимые внутренние стимулы. Что действительно может дать такие стимулы, так это ощущение значительности достигнутых результатов, гордость за хорошо выполненную работу, более высокая ответственность, продвижение по службе и профессиональный рост — все то, что обогащает работу



Личность человека раскрывается через четыре компонента — тело, сердце, разум и душу:
телу необходимы деньги и уверенность в завтрашнем дне
сердцу — любовь и признание
разуму — развитие и самосовершенствование
душе — самореализация

- ▶ Характеристики рабочей атмосферы в команде – дизайн помещения, оборудование рабочего места и предоставляемые сервисы, чистота, режим работы пр.
- ▶ Виды вознаграждений – оплата труда и другие выплаты, социальный пакет (система мед. обслуживания, возможность получить дополнительное образование и пр.), дополнительные материальные выгоды
- ▶ Микроклимат в коллективе – причастность к качественному выполнению проекта, уважение и одобрение коллектива, стиль общения с руководством, отношения между сотрудниками

Метод индивидуальной мотивации

Основан на изучении психологического портрета сотрудника, выявления его доминирующей потребности и предложения ему должности, где его потребность будет удовлетворена с наибольшей пользой для организации

Формула Врума:

$$M = a \times b \times c$$

a – личные ожидания по связи «затраты труда – результат»

b – личные ожидания по связи «результат – вознаграждение»

c – степень личной удовлетворенности результатом

Работник должен быть уверен в том, что если он будет прилагать усилия, то он сможет выполнить задание (величина a), причем за это ему хорошо заплатят (величина b) и это будет очень хорошо для него лично (величина c)



Виды вознаграждений

► **Внутренние – ценности, существующие в сознании человека:**

- чувство самоуважения
- удовлетворенность результатом
- ощущение значимости и ответственности своего труда
- комфорт неформального общения в коллективе

► **Внешние – ценности, предоставляемые организацией за выполненную работу:**

- зарплата
- премии
- продвижение по службе
- символы статуса и престижа
- похвалы и признания
- дополнительные льготы и вознаграждения

Потребности	Предпочтения сотрудников в зависимости от профессионального уровня, %		
	Начинающий	Опытный	Мастер
Материальные (зарплата, условия труда, социальный пакет)	50	20	-
Безопасность (стабильность компании, востребованность)	-	20	-
Принадлежность к команде (признание в коллективе, возможность учиться у более опытных коллег)	40	20	10
Самоуважение (профессиональный и карьерный рост, самостоятельность и ответственность в работе)	10	30	40
Самоактуализация (амбициозность целей)	-	10	50

Качества, повышающие ценность программиста как члена команды проекта

Как член команды программного проекта программист должен:

- ▶ занимать активную позицию, стремиться расширить свою ответственность и увеличивать личный вклад в общее дело
- ▶ постоянно приобретать новые профессиональные знания и опыт, выдвигать новые идеи, направленные на повышение эффективности реализации проекта, добиваться распространения своих знаний, опыта и идей среди коллег
- ▶ получать удовольствие от своей работы, гордиться ее результатами и стремиться, чтобы эти же чувства испытывали все коллеги
- ▶ четко осознавать свои личные и общие цели, понимать их взаимообусловленность, настойчиво стремиться к их достижению
- ▶ быть уверенным в себе и в своих коллегах, объективно оценивать их достижения и успехи, внимательно относиться к их интересам и мнениям, активно искать компромиссные решения в конфликтах
- ▶ всегда оставаться оптимистом, но при этом твердо знать, что окружающий мир несовершенен
- ▶ воспринимать каждую новую проблему как дополнительную возможность подтвердить собственный профессионализм





Спасибо за внимание!