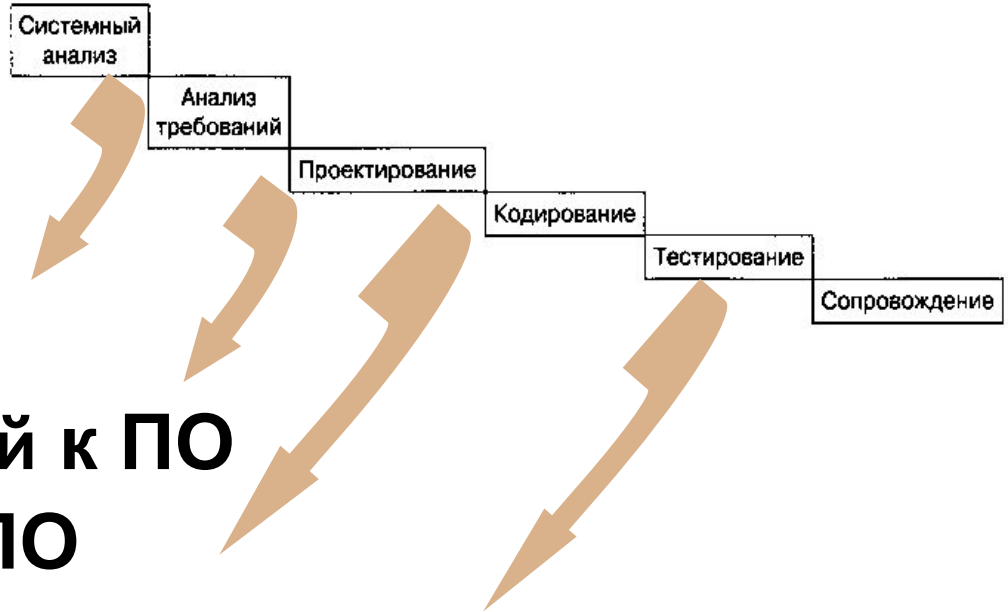
Технология программирования
(Software engineering)

Тема: Процессы программного обеспечения

доцент к.т.н. **Шалфеева**
Елена Арефьевна
shalf@dvo.ru

План изучения дисциплины

- **Процессы программного обеспечения**



- **Системная инжене рия** (начальные этапы)
- **Анализ требований к ПО**
- **Проектирование ПО**
- **Кодирование ПО и Испытания**
- **Управляющие и поддерживающие процессы**

ИАН СОММЕРВИЛЛ

Инженерия программного обеспечения

6-е издание

Addison-Wesley

ВИЛЬЯМС

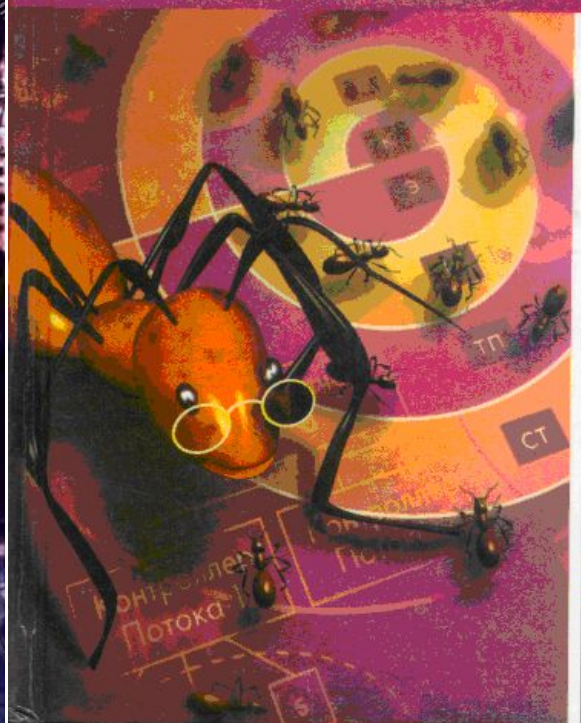
С. А. Орлов

ПИТЕР

ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

РАЗРАБОТКА СЛОЖНЫХ ПРОГРАММНЫХ СИСТЕМ

УЧЕБНИК ДЛЯ ВУЗОВ



- для студентов и преподавателей высших учебных заведений направлени «Информатика и вычислительная техника»
- фундаментальный курс по программной инженерии

РЕКОМЕНДОВАНО МИНИСТЕ ОБРАЗОВАНИЯ РФ

Роль программного обеспечения

Ранние годы

первое поколение компьютеров,
программирование в машинных кодах.
Ориентация на Пакет,
Ограниченное Распространение,
ПО для конкретных заказчиков,
Интуитивная технология программирования

Вторая эра

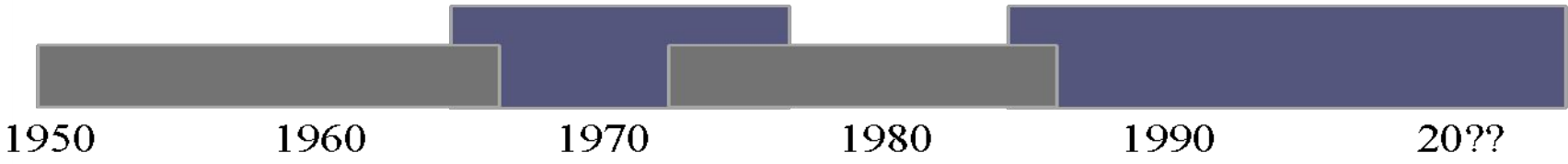
Многопользовательский режим,
Реальное время,
Большие затраты на Компиляцию,
Базы данных,
ПО как продукт на продажу,
Сопровождение программного, обеспечения.
Кризис программного обеспечения

Третья эра

Появление микропроцессоров и встроенная интеллектуальность
Распределенные системы;
внимание к тестированию ПО;
попытка введения метрик;
Стандартизация БД,
реляционный подход к СУБД.
Интерес к CASE-инструментарию

Четвертая эра

Мощные настольные системы,
Параллельные Вычисления,
Сетевые ЭВМ,
Объектно-Ориентированные Технологии,
Искусственный интеллект,
Модель Capability Maturity Model.



Эволюция программного обеспечения и технологии.

Эволюция программного обеспечения и инженерии

Третья эра
встроенная интеллектуальность,
распределенные системы;
стандартизация БД,
интерес к CASE-инструментарию…

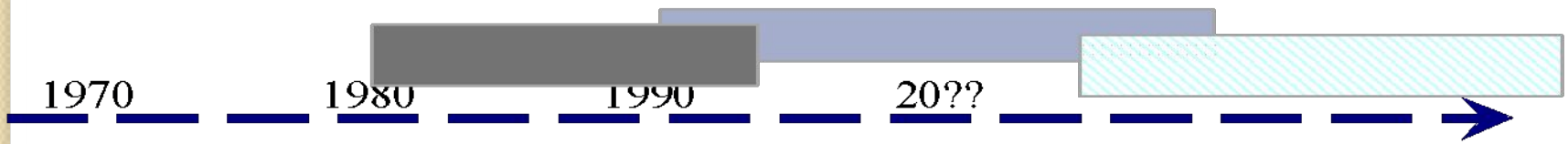
Четвертая эра
Мощные настольные системы,
Параллельные Вычисления,
Сетевые ЭВМ,
Объектно-Ориентированные Технологии,
Искусственный интеллект,
Модель Capability Maturity Model.

?пятая?

носимая электроника?

ДНК-компьютер?

Wetware?



? «PC+ эра» (мнение компании Microsoft)

Молодая, но уже большая часть ИТ-индустрии - устройства, отличные от PC. Такие как смартфоны и **носимая электроника**: умные часы и фитнес браслеты.

Носимая электроника способствует развитию искусственного интеллекта и новых способов взаимодействия вычислительных машин с людьми: *распознавание образов, речи, жестов*.

В наше время ведущие производители электронных устройств перестали рекламировать вычислительные способности своих продуктов, а ставят на первый план интерфейс и различные **интеллектуальные возможности** своих устройств.

? Wetware

- ***Искусственная нейронная сеть*** — программная или аппаратная реализация организации и функционирования сетей нервных клеток живого организма.
- К настоящему моменту накоплено большое число различных «правил обучения» и архитектур нейронных сетей, их аппаратных реализаций и приёмов использования нейронных сетей для решения прикладных задач.
- «Созревающее» новое направление а нейрокомпьютинге основано на соединении биологических нейронов с электронными элементами. По аналогии с Software и Hardware, эти разработки получили наименование Wetware («влажный продукт»).
- В настоящее время уже существует технология соединения биологических нейронов со сверхминиатюрными полевыми транзисторами с помощью нановолокон (Nanowire). В разработках используется современная нанотехнология. В том числе, для создания соединений между нейронами и электронными устройствами используются углеродные нанотрубки.

? квазибиологическая парадигма

Биокомпьютер (индивидуальная машина) – противоположность универсальному компьютеру «фон Неймана».

Начиная с 2002 года исследователи Института Вейцмана (Израиль) представляли работы на эту тему. И в апреле 2004 года сообщили о создании **ДНК-компьютера** с модулем ввода-вывода данных (способен диагностировать опухоли на клеточном уровне и выпускать противораковые препараты).

В январе 2013 года смогли записать в ДНК-коде несколько фотографий и звуковых файлов.

В Стэнфорде создан биологический транзистор (март 2013)– Транскриптор. Биологический транзистор внутри живой клетки способен менять состояние под действием внешнего стимула. По словам участников проекта, использование биокомпьютеров позволит изучать и «перепрограммировать» живые системы.

«Пятое поколение» развивается на исследовательском уровне.

Факторы, способствующие появлению и развитию SE

- Почему необходимо так **много времени для завершения** разработки программы?
- Почему ее **стоимость столь высока**?
- Почему мы не можем **найти все ошибки** до то, как мы передадим программу нашим заказчикам?

Проблемы, относящиеся к ПО

- **"кризис программирования":** программные проекты (Projects) часто отменяются до своего завершения, а завершенные проекты часто выходят за рамки бюджета и сроков, причем их результаты имеют невысокое качество;

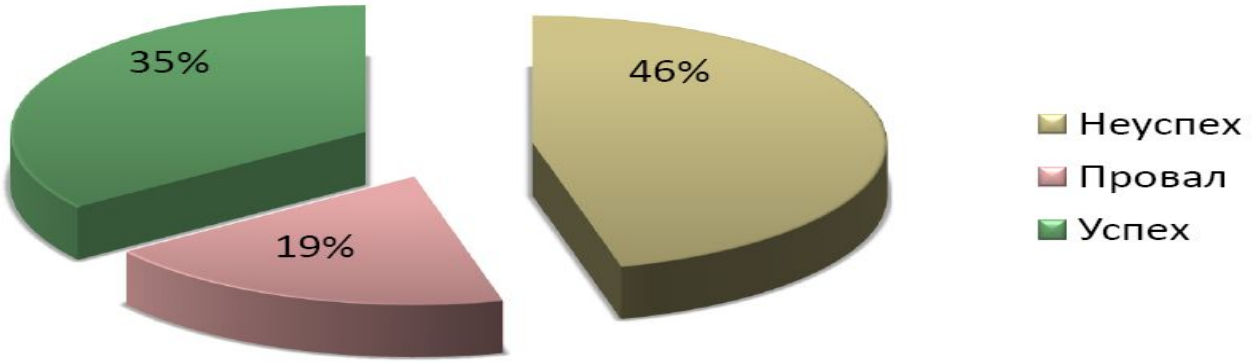
«Программные проекты часто отменяются до своего завершения, а завершенные проекты часто выходят за рамки бюджета и сроков, причем их результаты имеют невысокое качество.» [**Кони Бюрер (*Rational Software*)**] .

Факторы, способствующие развитию ТРПО

Продолжают существовать и усиливаться множество относящихся к ПО проблем:

- Повсеместное использование компьютеров сделало общество все более зависимым от **надежной работы ПО**. Чрезвычайные экономические потери и потенциальные человеческие страдания могут произойти в случае сбоя ПО. (расходы из-за некачественного ПО также увеличиваются)
- Сегодня программное обеспечение – область с чрезвычайно сильной **конкуренцией**. Разработанное программное средство может продаваться многократно в готовом виде. **Стоимость, график и качество становятся основными характеристиками,** определяющими напряженную конкуренцию за программную работу.

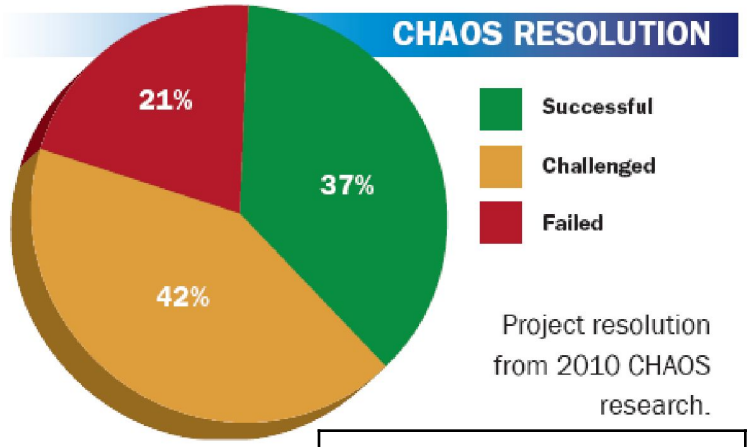
Статистика производства ПО

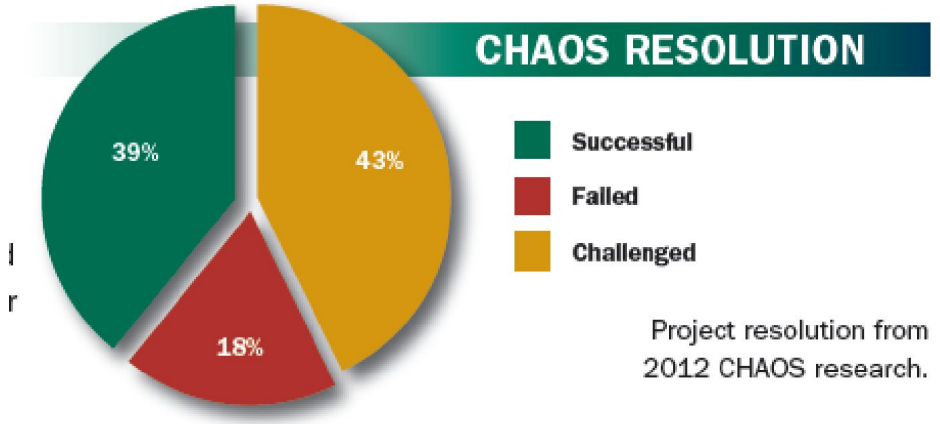


Источник: "The Chaos Report" The Standish Group, 2008

- CHAOS research encompasses 18 years of data on why projects succeed or fail, representing **more than 90,000 completed IT projects.** However, for our new database we eliminated cases from 1994 though 2002, since they did not match the current requirements for analysis. The new database has **just under 50,000 projects.**
- CHAOS results provide a global view of project statistics but do tend to have a heavier concentration on the United States and Europe. For each reporting period, about 60% of the projects are U.S. based, 25% are European, and the remaining 15% represent the rest of the world.

Статистика *The Standish Group* по программным проектам





Год	Проваленные	Проблемные	Успешные
2012	18%	43%	39%
2010	21%	42%	37%
2008	24%	44%	32%
2006	19%	46%	35%
2004	18%	53%	29%
2002	15%	51%	34%
2000	23%	49%	28%
1998	28%	46%	26%
1995 (23 тыс.)	31,1	52,7	16,2%
80e			14%

В 2014

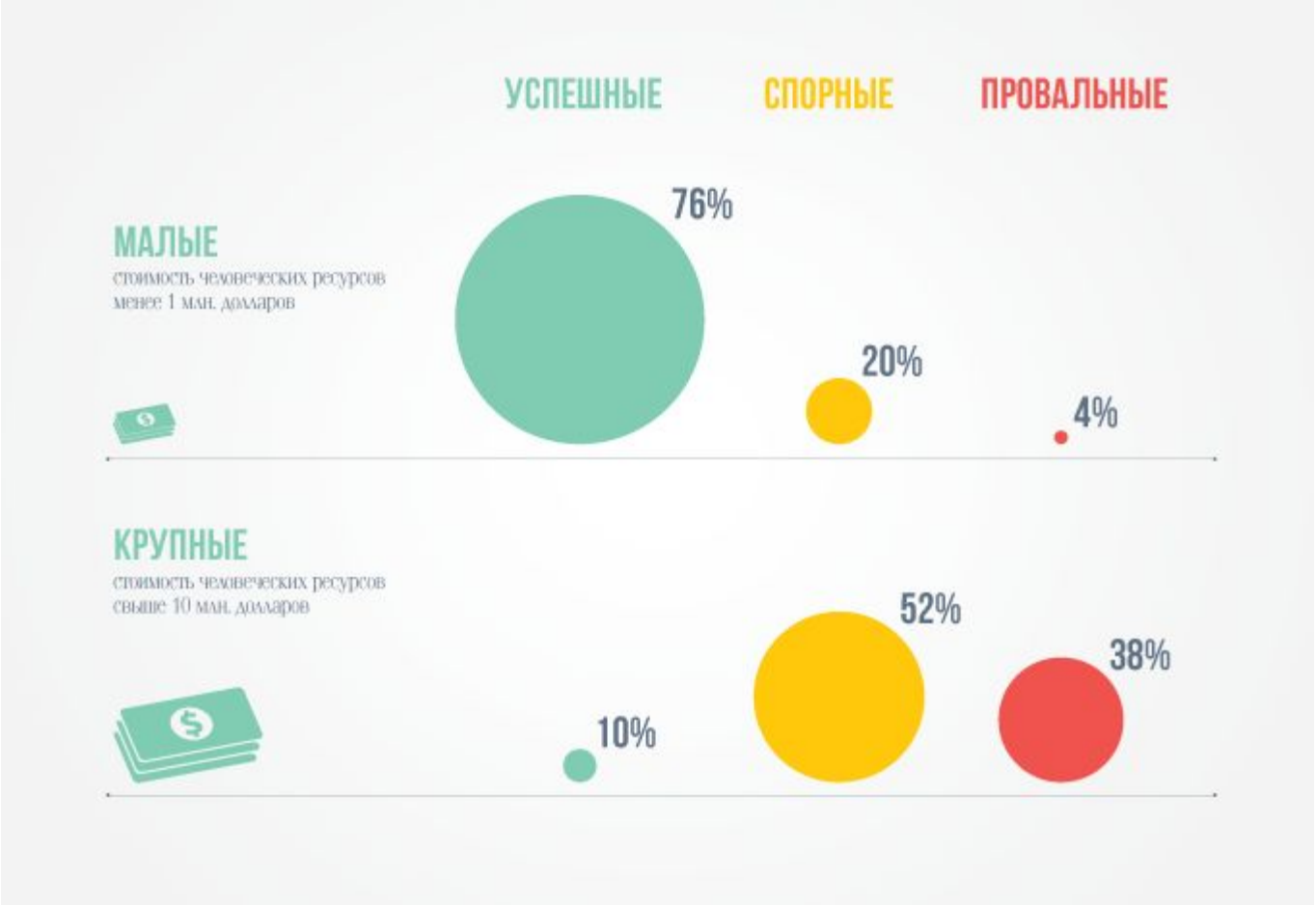
- по данным The Standish Group International, в 2014 году 52,7% ИТ- проектов столкнулись во время разработки с проблемами, которые оказали значительное влияние на длительность, бюджет, качество и впоследствии привели к изменению ранее запланированных целей и ожидаемых результатов. Порядка 31,1% проектов были остановлены и не завершены.

	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%

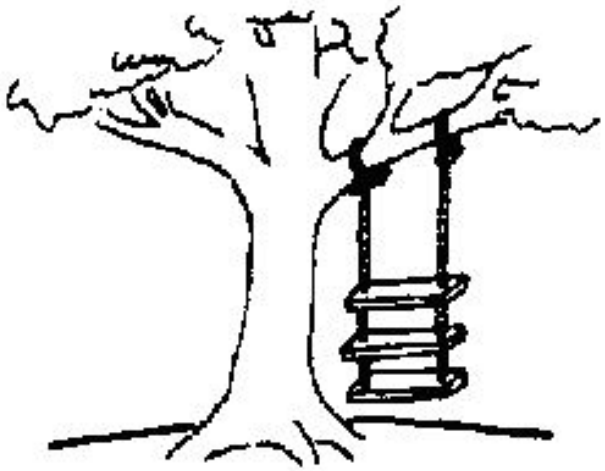
Зависимость успеха от масштаба проекта

Статистика The Standish Group демонстрирует зависимость успеха от масштаба проекта:

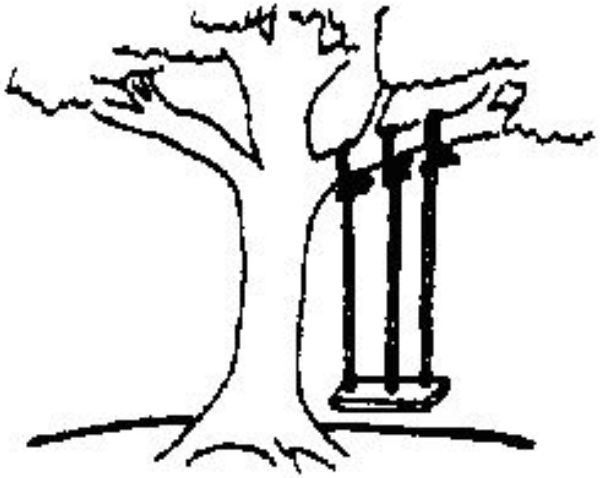
По небольшому проекту прогноз сроков окажется более достоверным [The Standish Group. CHAOS MANIFESTO 2013]



Как обычно пишутся программы

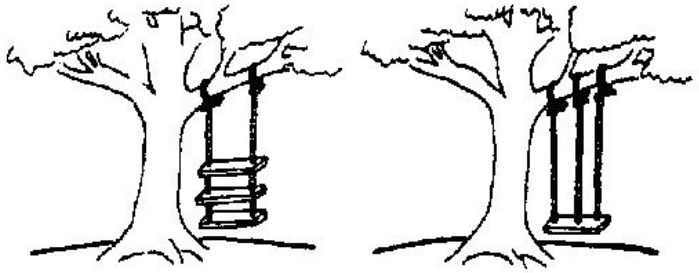


ТАК БЫЛО ПОСТАВЛЕНО ТЕХНИЧЕСКОЕ ЗАДАНИЕ



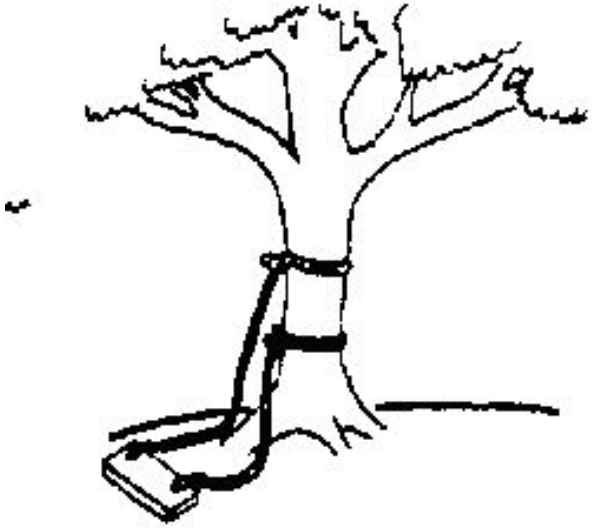
ТАК ЕГО ПОНЯЛИ РАЗРАБОТЧИКИ

Как обычно пишутся программы

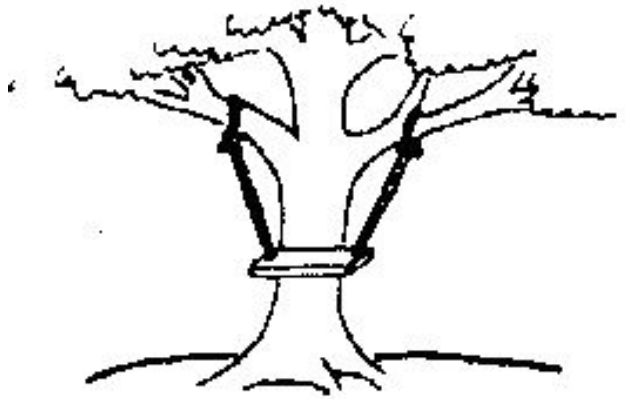


ТАК БЫЛО ПОСТАВЛЕНО ТЕХНИЧЕСКОЕ ЗАДАНИЕ

ТАК ЕГО ПОНЯЛИ РАЗРАБОТЧИКИ

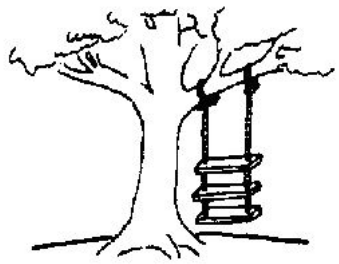


И ТАК ЭТУ ЗАДАЧУ И РЕШАЛИ РАНЬШЕ

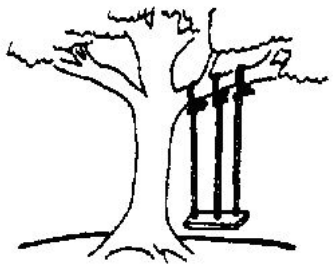


ТАК ЕЕ РЕШИЛИ ТЕПЕРЬ

Как обычно пишутся программы

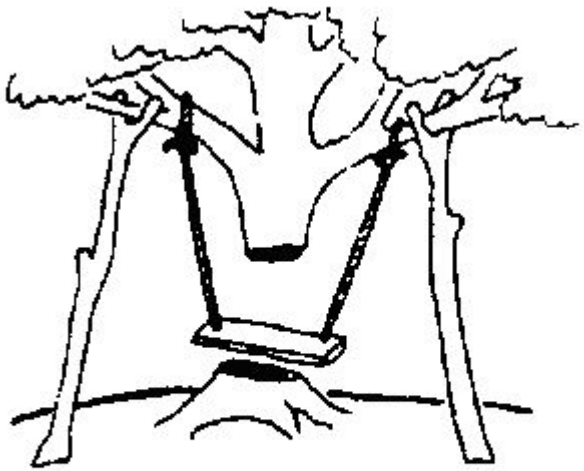


ТАК БЫЛО ПОСТАВЛЕНО ТЕХНИЧЕСКОЕ ЗАДАНИЕ

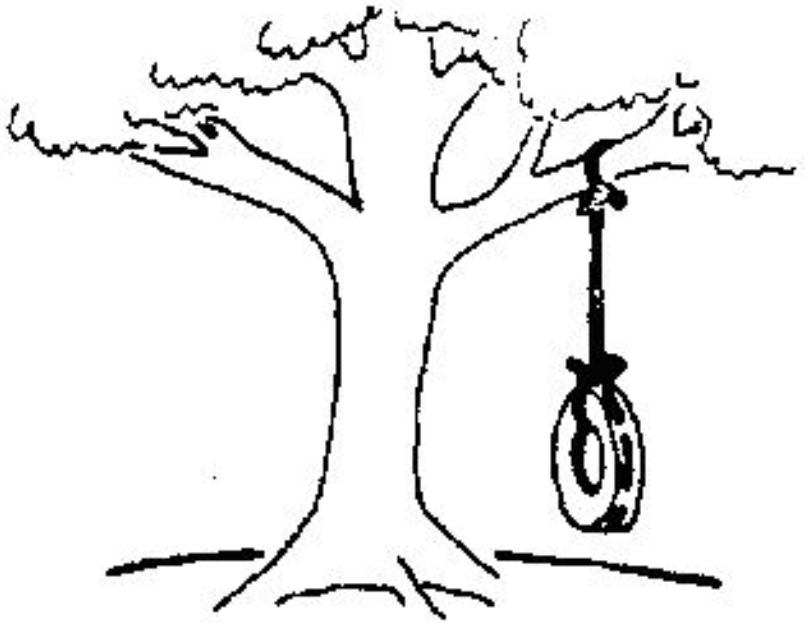


ТАК ЕГО ПОНЯЛ РАЗРАБОТЧИК

ТАК ЕЕ РЕШИЛИ ТЕПЕРЬ



ТАКОЙ ПРОГРАММА СТАЛА ПОСЛЕ ОТЛАДКИ



А, СОБСТВЕННО, ТАК ЕЕ ПРЕДСТАВЛЯЛ СЕБЕ ЗАКАЗЧИК

Программное обеспечение

- Программное обеспечение это:
- 1) команды (компьютерные программы), которые при выполнении обеспечивают желаемые функции и характеристики;
- 2) структуры данных, которые дают возможность программам адекватно манипулировать информацией; и
- 3) документы, которые описывают работу и использование программ.

Некоторые характеристики программного обеспечения

- Программное обеспечение разрабатывается или конструируется, оно не производится в классическом понимании

  Кони Бюрер (*Rational Software*): «…Граница между проектированием и строительством всегда четко обозначена артефактом – **чертежом.** Проектирование включает в себя все операции, необходимые для создания чертежа, а строительство охватывает все операции, необходимые для создания продуктов по этому чертежу. **В идеальном случае чертеж должен определять создаваемый продукт во всех подробностях – *все подробности программной системы описываются только кодом*** на языке высокого уровня, который, таким образом**, является чертежом программы.**

- Программный продукт нематериален;
- Программное обеспечение не изнашивается;
- Послепроизводственные проблемы решаются «более просто»;
- Большая часть программного обеспечения разрабатывается для конкретных заказчиков, а не собирается из готовых компонентов.

Классы программного обеспечения

- ***Заказной (custom)*** **ПП - для конкретного заказчика, малый (часто единичный) тираж.**
- ***Рыночный (commercial)*** **ПП - для широкого круга пользователей, неограниченный тираж.**

Приложения программного обеспечения (в зависимости от их информационного содержания и детерминированности)

- *Системное* программное обеспечение
- *Инженерное и научное* программное обеспечение
- *Встроенное* программное обеспечение
- Программное обеспечение *реального времени*
- Программное обеспечение *искусственного интеллекта*
- ***Системы делопроизводства, Бухгалтерские, Финансовые***
- ***Системы*** и прочее *Коммерческое* программное обеспечение
- ***Текстовые (и другие) редакторы*** и прочее программное обеспечение *персональных ЭВМ*
- *Программы -клиенты для электронной почты, блогов* и прочие *программы для удаленного взаимодействия\общения*

Точки зрения на предмет *технологии программирования*:

- (1) широкое использование инструментальных средств или
- (2) набор методик и регламентирующих средств, позволяющих, в частности, на каждом этапе провести экспертизу, архивацию и измерение объема и качества проделанной работы (средства – *стандарты*, *процедуры*: соглашения о последовательности действий одного или нескольких разработчиков, направленных на достижение некоторой цели ТП, *программные средства*), или
- (2) *"наука и искусство"* (в качестве средств – *"методы жизненного цикла").*

Технология программирования (современный взгляд)

Сейчас технология программирования это – *"инженерная дисциплина"* с *"инженерными подходами"*. В качестве ее **средств** рассматриваются *"стандарты, методологии, технологические программные средства, процессы, методы организации, методы управления и системы обеспечения качества"*.

Основной принцип, поддерживающий технологию программирования как инженерную дисциплину, это фокусирование внимания на **качестве**. Любое применение технологии программирования, означает взятие организацией-производителем обязательств по выпуску продукции "высокого" качества.

Технология программирования

Согласно американскому стандарту *ANSI/IEEE 610.12-1990:*

"Технология программирования - это:

- *1) Применение систематического, упорядоченного, поддающегося количественному определению подхода для разработки программного обеспечения, работы с ним и его сопровождения, то есть, применение инженерного дела к программному обеспечению;*
- *2) Дисциплина, изучающая подходы (1)."*

Понятие процесса

- *Технологический* ***процесс*** **(process) - элемент в структуре ЖЦ ПО \ПС** — *множество взаимосвязанных видов деятельности*, ***решающих некоторую общую задачу*** *(problem) или связанную совокупность задач* (напр, *анализ требований, проектирование системной архитектуры, процесс сопровождения ПО, процесс обеспечения качества, процесс разработки документации и* ***пр.***).
- ***Деятельность*** **(activity)** - часть *процесса*, выполняемая одним человеком или группой, ***преобразующими получаемую ими входную информацию в выходную*** (получают некоторую информацию, передаваемую устно, в виде документов, программ и т. д. от предыдущего шага и на ее основе создают выходную информацию (представляемую в различной форме).
  - Каждая **деятельность** представляется набором инженерных рабочих задач.
- ***Задачи*** ***(Tasks)*** атомарны (выполняются обычно одним человеком) и состоят в выполнении некоторого действия.

Вид деятельности	Выходные документы
Анализ требований	Исследование реализуемости Предварительные требования
Определение требований	Документ с требованиями
Спецификация системы	Спецификация функций План приемочных испытаний Предварительное руководство пользователя
Архитектурное проектирование	Спецификация архитектуры План испытаний системы
Проектирование интерфейса	Спецификация интерфейса План испытаний при комплексировании
Детальное проектирование	Спецификация проекта План испытаний программных единиц
Кодирование	Код программы
Испытание программных единиц	Отчет об испытаниях программных единиц
Испытания при комплексировании	Отчет об испытании при комплексировании Окончательное руководство пользователя
Испытание системы	Отчет об испытаниях системы
Приемочные испытания	Окончательный вариант системы и системная документация

деятельность «Архитектурное проектирование

Проектирование программной архитектуры состоит из следующих задач:

- *трансформировать требования к программному объекту в архитектуру*, которая описывает общую структуру; объекта и определяет компоненты программного объекта.
- *разработать и документально оформить общий (эскизный) проект базы данных;*
- *определить и документально оформить предварительные общие требования к испытаниям* программного объекта и график сборки программного продукта.
- *оценить архитектуру* программного объекта по следующим критериям:
  - a) учет требований к программному объекту;
  - b) внешняя согласованность с требованиями к программному объекту;
  - c) внутренняя согласованность между компонентами программного объекта;
  - d) соответствие методов проектирования и используемых стандартов;
  - e) возможность технического проектирования.

деятельность «Аттестация»

состоит из следующих ***задач:***

***Подготовка выбранных требований к испытаниям*, контрольных примеров** и технических условий испытаний **к анализу результатов испытаний**.

Проведение испытаний, включая:

- a) **испытания при критических, граничных и особых значениях исходных данных**;
- b) **испытание программного продукта на способность изолировать и минимизировать эффект ошибок** с постепенным понижением влияния сбоев и запросом помощи оператора при критических, граничных и особых условиях;
- c) **испытание при участии репрезентативно выбранных пользователей**, могущих успешно решать свои задачи при использовании данного программного продукта.

***Подтверждение того, что программный продукт удовлетворяет* заявленным возможностям.**

«слои» технологии программирования

Методы ТП представляют собой структурные **решения, предназначенные для разработки высококачественного ПО эффективным способом**, они задают техническое описание того, "как сделать", чтобы построить программное средство или его компонент. Они включают в себя ***правила моделирования, формализованную нотацию***, другие наглядные приемы, а также способы управления процессом создания ПО.

Слои технологии программирования.

Средства (tools) ТП обеспечивают автоматическую или полуавтоматическую поддержку процессов и методов. Это ***системы программирования, инструментальные программные средства*** (toolkits, CASE-средства), аппаратура (компьютеры, сети, ...)

CASE - *автоматизированная* или *поддерживаемая ЭВМ технология программирования* (computer-aided software engineering) - программные системы для автоматизации процесса создания ПО, в которых информация, создаваемая одним из средств, используется другим.

Обобщенный взгляд на технологию программирования

- **Фазы определения** (**какие** функции, информация, «поведение»…)**,**
- **разработки** (**как** построить) и
- **сопровождения** (исправление, адаптация, модернизация, предотвращение)

(Р. Боровко / CNews Analytics): «Согласно отраслевым исследованиям, компании, разрабатывающие ПО тратят 38% своих времени и денег на исправление уже написанного кода…»

Общая схема процессов (common process framework)

устанавливается определением небольшого числа *видов деятельности схемы*, которые применимы ко всем проектам ПО, независимо от их размера или сложности.

- **Системная/информационная инженерия**
- **Анализ требований к программному обеспечению**
- **Проектирование**
- **Генерация кода**
- **Испытания**
- **Сопровождение программного средства**

Виды деятельности
Анализ требований
Определение требований
Спецификация системы
Архитектурное проектирование
Проектирование интерфейса
Детальное проектирование
Кодирование
Испытание программных единиц
Испытания при комплексировании
Испытание системы
Приемочные испытания

Пропорции стадий этапа разработки

*[*В. Иванов, «Руководство по управлению внедренческими **проектами на базе MS PROJECT 2000** и рекомендаций PMI»*)] Этап разработки разделяется на стадии :*

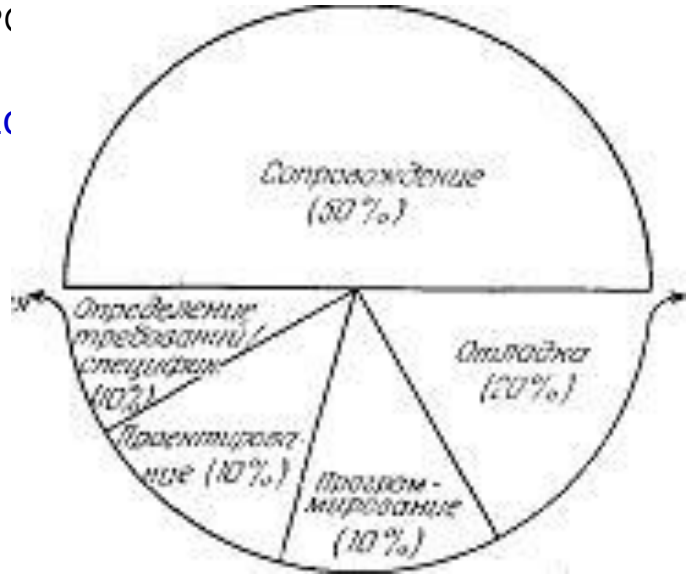
- *-**Постановка - 34%***
- ***-Кодирование - 21%***
- *-**Тестирование - 45%***

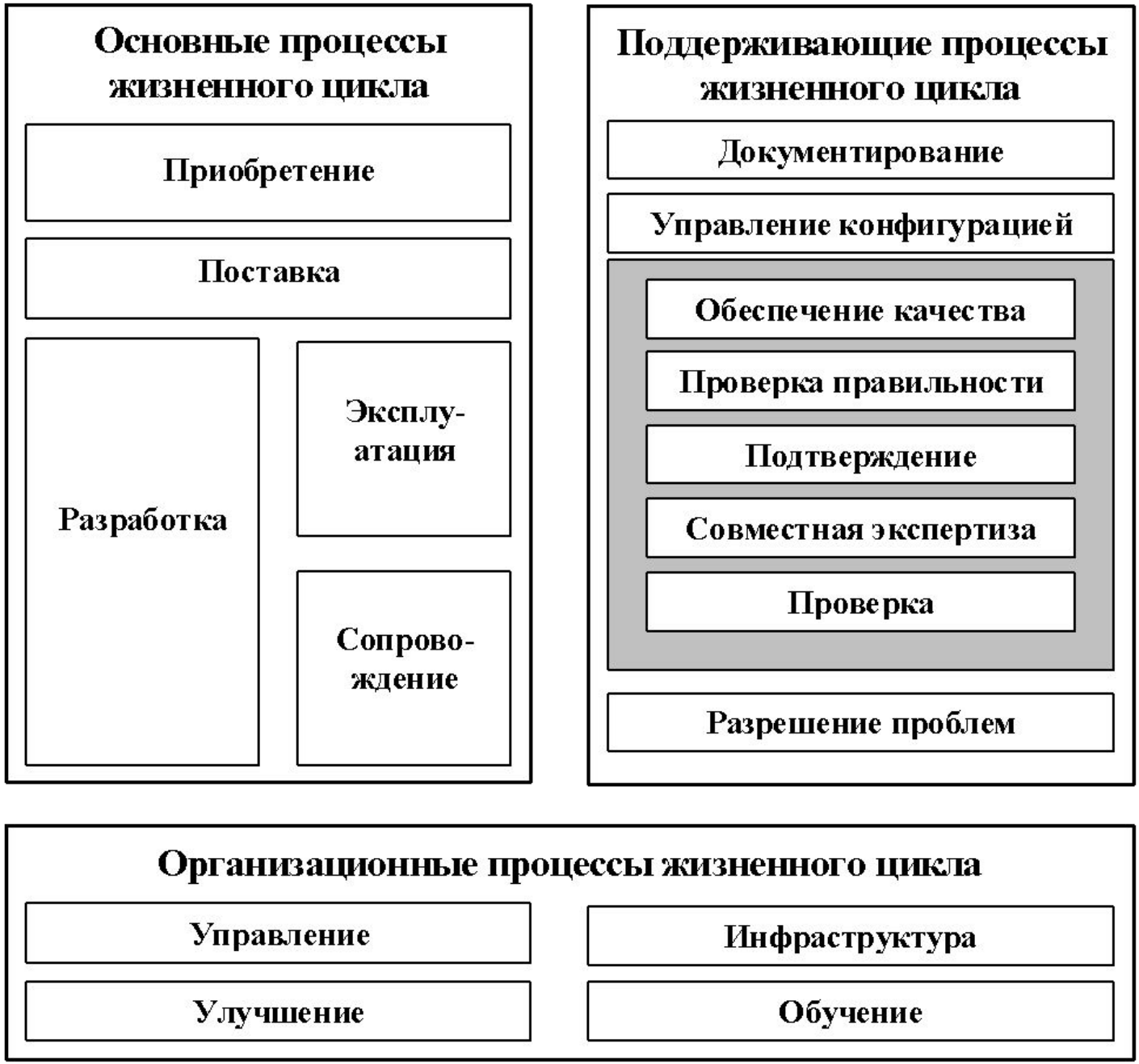
Брукс: *В течение ряда лет … пользуюсь следующим эмпирическим правилом:*

- ***1/3 - планирование,***
- *1/6 - написание программ,*
- ***1/4 - тестирование** компонентов и предварительное системное тестирование,*
- ***1/4 - системное тестирование** при наличии всех компонентов.*

(По материалам **Р.Гласс, Р.Нуазо**, "Сопровождение ПР(
1983, Москва «Мир»):

- *Определение и специфик-я требов-й – 1(*
- ***Проектирование – 10%;***
- *Программирование – 10%;*
- ***Отладка – 20%;***
- ***Сопровождение – 50%.***





Программные процессы стандарта *ISO/IEC 12207:1995*.

Основные процессы

Основные процессы (primary life cycle processes)

- процессы, которые **реализуются главными участниками жизненного цикла программного средства.**

Главный участник – это тот, кто инициирует или осуществляет разработку, эксплуатацию или сопровождение программного обеспечения: покупатель, поставщик, разработчик, оператор и специалист по сопровождению программных продуктов.

Процесс **разработки** (development)

определяет действия разработчика, организации, которая создает модель программного продукта, а затем разрабатывает его;

состоит из следующих **видов деятельности** (activities):

- ***Развертывание процесса разработки.***
- ***Анализ требований к системе.***
- Создание архитектурного проекта (System architectural design) системы
- Анализ требований к ПО.
- Создание архитектурного проекта (**дизайна**) ПО.
- Создание детального проекта (**дизайна**) ПО.
- Кодирование и тестирование ПО.
- Сборка ПО (Software integration).
- Квалификационное тестирование ПО.
- Сборка (**интеграция**) системы.
- Квалификационное тестирование системы (System qualification testing).
- Установка ПО.
- Поддержка заказчика при проведении приёмки ПО.

Развертывание процесса разработки

Развертывание процесса разработки состоит **из задач:**

- *определения модели жизненного цикла, соответствующей области реализации, величине и сложности проекта. При этом в ней должны быть выбраны и структурированы работы и задачи* ***процесса разработки;***
- *выбора используемых стандартов, языков и инструментов*, которые документально оформлены и приняты в организации разработчика;
- *разработки планов проведения работ в процессе разработки* (Планы должны охватывать конкретные стандарты, методы, инструментарий, действия и обязанности, связанные с разработкой и квалификацией всех требований, включая безопасность и защиту).

Анализ требований к системе

Данная работа состоит из следующих **задач,** которые разработчик должен выполнить или обеспечить их выполнение:

5.3.2.1 Разработчик, при необходимости, должен ***выполнить анализ области применения разрабатываемой системы с точки зрения определения требований к ней.*** Технические требования к системе должны охватывать: *функции и возможности системы; коммерческие и организационные требования; требования пользователя; требования безопасности и защиты; эргономические требования; требования к интерфейсам; эксплуатационные требования; требования к сопровождению; проектные ограничения и квалификационные требования*. Технические требования к системе должны быть документально оформлены.

5.3.2.2 ***Требования к системе должны быть оценены*** с учетом следующих критериев (при этом результаты оценок должны быть документально оформлены):

- a) учет потребностей заказчика;
- b) соответствие потребностям заказчика;
- c) тестируемость;
- d) выполнимость проектирования системной архитектуры;
- e) возможность эксплуатации и сопровождения.

Процесс приобретения (acquisition)

Процесс приобретения определяет действия покупателя - организации, которая приобретает систему, программный продукт или программное обслуживание; он состоит из следующих видов деятельности:

- *Инициация.*
- *Подготовка запроса с предложением (тендера).*
- *Подготовка и обновление контракта.*
- *Мониторинг поставщика.*
- *Приёмка готовой системы или ПО и завершение работы.*

Поддерживающие процессы

Поддерживающие процессы жизненного цикла (supporting life cycle processes).

Поддерживающий процесс **поддерживает другой процесс** как неделимую часть с различными целями и **вносит определенный вклад в успех и качество программного проекта** (project).

Поддерживающий процесс запрашивается и выполняется, если это необходимо, другим процессом.

Поддерживающие процессы:

- *процесс документирования* (documentation).
- *процесс управления конфигурацией* (configuration management).
- *процесс обеспечения качества* (quality assurance),
- *процесс проверки правильности (верификация)* (verification).
- *процесс подтверждения* (validation).
- *процесс совместной экспертизы*
- *процесс проверки* (audit).
- *процесс разрешения проблем* (problem resolution).

Организационные процессы

Организационные процессы (organizational life cycle processes) связаны с управлением, созданием необходимой инфраструктуры, обучением персонала, внесением улучшений в процесс. **Они задействуются организацией для установления и реализации основной структуры, полученной объединенными процессами жизненного цикла и персонала, и для непрерывного улучшения этой структуры и процессов.**

Они обычно задействуются извне сферы конкретных проектов или контрактов.

Организационные процессы – это:

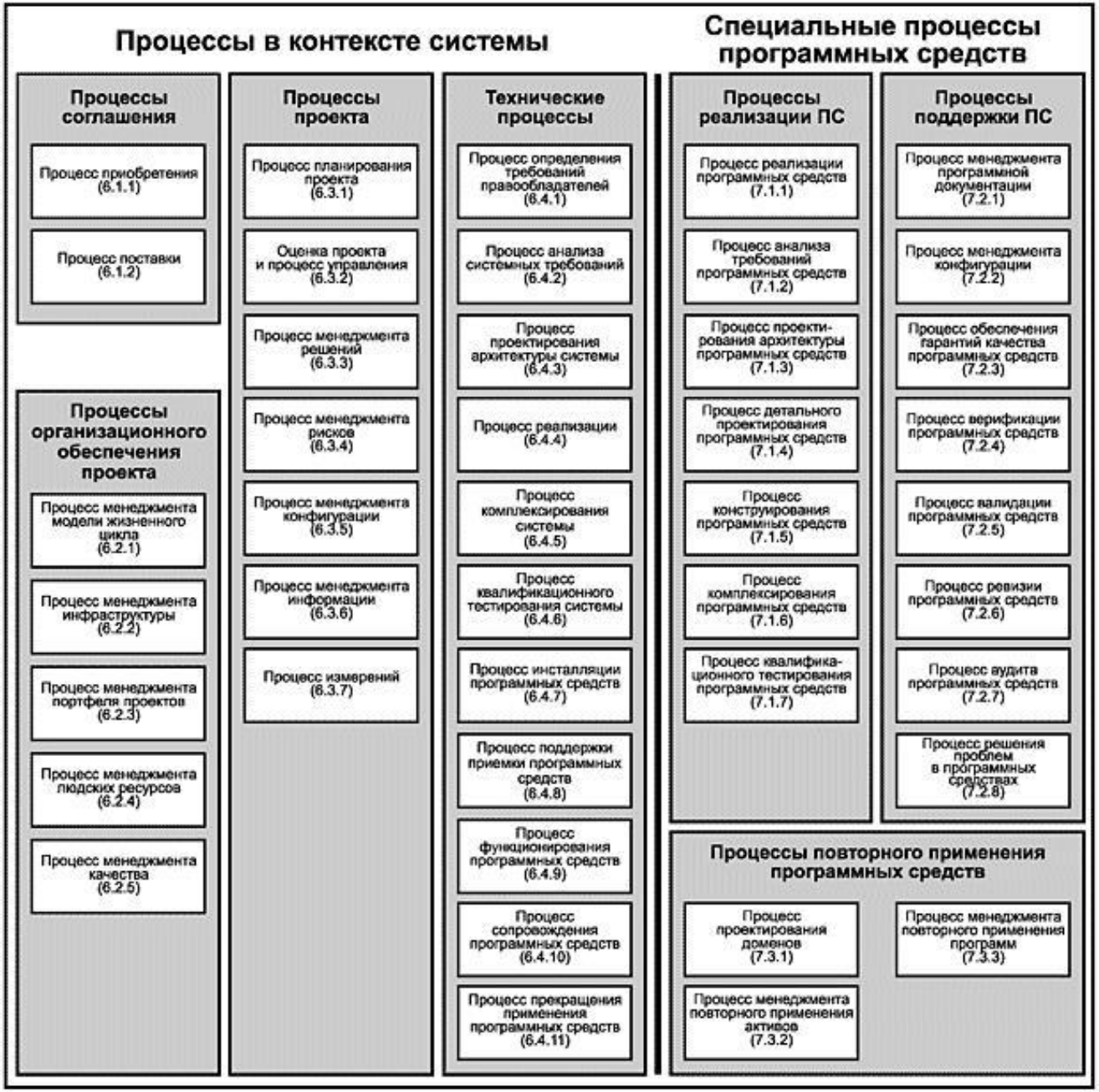
- 1) *процесс управления* (management);
- **2)** *процесс установления инфраструктуры* (infrastructure);
- 3) *процесс улучшения* (improvement);
- 4) *процесс обучения* (training).

ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия

- Настоящий стандарт группирует различные виды деятельности, которые могут выполняться в течение жизненного цикла программных систем, в семь групп процессов.
- a) процессы соглашения - два процесса (см. 5.2.2.1.1 и 6.1);
- b) процессы организационного обеспечения проекта - пять процессов (см. 5.2.2.1.2 и 6.2);
- c) процессы проекта - семь процессов (см. 5.2.2.1.3 и 6.3);

  d) технические процессы - одиннадцать процессов (см. 5.2.2.1.4 и 6.4);
- e) процессы реализации программных средств - семь процессов (см. 5.2.2.2.1 и 7.1);
- f) процессы поддержки программных средств - восемь процессов (см. 5.2.2.2.2 и 7.2);

Группы процессов жизненного цикла



Каждый из процессов жизненного цикла в пределах этих групп описывается в терминах цели и желаемых выходов, списков действий и задач, которые необходимо выполнять для достижения этих результатов.

- **6.4 Технические процессы**
- 6.4.1 Процесс определения требований правообладателей
- 6.4.2 Процесс анализа системных требований
- 6.4.3 Процесс проектирования архитектуры системы
- 6.4.4 Процесс реализации
- 6.4.4.1 Цель

  Цель процесса реализации заключается в создании заданных элементов системы.

  Примечание - Пользователи настоящего стандарта могут иметь намерение работать с программными продуктами или программными элементами больших систем. Процесс реализации программных средств (см. 7.1.1) является соответствующим примером процесса реализации в [18], приспособленного к частным потребностям реализации программного продукта или услуги.
- **6.4.5 Процесс комплексирования системы**

- **7 Процессы жизненного цикла программных средств**
- **7.1 Процессы реализации программных средств**

- 7.1.1 Процесс реализации (является частным случаем процесса реализации из [18], приспособленного к специфическим потребностям реализации программного продукта или услуги.

7.1.1.1 Цель заключается в **создании заданных элементов системы, выполненных в виде программных продуктов или услуг**.
В ходе этого процесса происходит преобразование заданных поведенческих, интерфейсных и производственных ограничений в действия, которые создают системный элемент, выполненный в виде программного продукта или услуги, известный как "программный элемент".

Результатом процесса является создание программной составной части, удовлетворяющей как требованиям к архитектурным решениям, что подтверждается посредством верификации, **так и требованиям правообладателей**, что подтверждается посредством валидации.

7.1.1.2 Выходы
a) определяется стратегия реализации;
b) определяются ограничения по технологии реализации проекта;

- c) изготавливается программная составная часть;
- d) программная составная часть упаковывается и хранится в соответствии с соглашением о ее поставке.

В дополнение к этим действиям процесс реализации программных средств имеет следующие процессы более низкого уровня:
- процесс анализа требований к программным средствам*;
- процесс проектирования архитектуры программных средств*;
- процесс детального проектирования программных средств;
- процесс конструирования программных средств;

7.1.2 Процесс анализа требований к программным средствам

Примечание - Процесс анализа требований к программным средствам в настоящем стандарте является процессом более низкого уровня, чем процесс реализации программных средств. Пользователи [18] могут решить, что этот процесс предусматривается процессом анализа требований [18] при рекурсивном применении [18].

7.1.2.1 Цель

Цель процесса анализа требований к программным средствам заключается в установлении требований к программным элементам системы.

7.1.2.2 Выходы

a) определяются требования к программным элементам системы и их интерфейсам;
b) требования к программным средствам анализируются на корректность и тестируемость;
c) осознается воздействие требований к программным средствам на среду функционирования;
d) устанавливается совместимость и прослеживаемость между требованиями к программным средствам и требованиями к системе;
e) определяются приоритеты реализации требований к программным средствам;
f) требования к программным средствам принимаются и обновляются по мере необходимости;
g) оцениваются изменения в требованиях к программным средствам по стоимости, графикам работ и техническим воздействиям;
h) требования к программным средствам воплощаются в виде базовых линий и доводятся до сведения заинтересованных сторон.

Международные стандарты ISO/IEC

Стандарт **ISO/IEC *12207:1995***

Всего определено 224 различные задачи.

Стандарт **ISO/IEC 15504** предназначен в основном для оценивания процессов.

Модель процессов стандарта ISO/IEC 15504 содержит 29 процессов, объединённых в **пять категорий: *заказчика-поставщика, инженерную, поддерживающую, менеджерскую и организационную***.

В отличие от ISO/IEC 12207, **процессы** в ISO/IEC 15504 состоят не из видов деятельности, а **сразу из атомарных задач** или практик (practice), как они называются в ISO/IEC 15504.

При этом задачи в ISO/IEC 15504 в целом соответствуют задачам из ISO/IEC 12207, т. е. на нижнем уровне модели процессов этих стандартов совпадают. В модели процессов стандарта ISO/IEC 15504 каждая задача может входить только в один процесс, а каждый процесс – только в одну категорию.

ISO/IEC 15504

Категория **процессов *Заказчика-поставщика***:

- *Приобретение ПО (Acquire software)*.
- *Управление потребностями заказчика (Manage customer needs)*.
- *Поставка ПО (Supply software)*.
- *Эксплуатация ПО (Operate software*).
- *Обеспечение поддержки заказчика (Provide customer service*.

***Инженерная* категория** процессов состоит из процессов, связанных с разработкой системы, ПО и документации к ним.

- *Разработка требований к системе и проекта (**дизайна**) (Develop system requirements and design)*.
- *Разработка требований к ПО (Develop software requirements)*
- *Разработка проекта (**дизайна**) ПО (Develop software design)*.
- *Выполнение проекта (**дизайна**) ПО (кодирование) (Implement software design)*.
- *Сборка (**интеграция**) и тестирование ПО (Integrate and test software)*.
- *Сборка (**интеграция**) и тестирование системы (Integrate and test system)*.
- *Сопровождение системы и ПО (Maintain system and software)*.

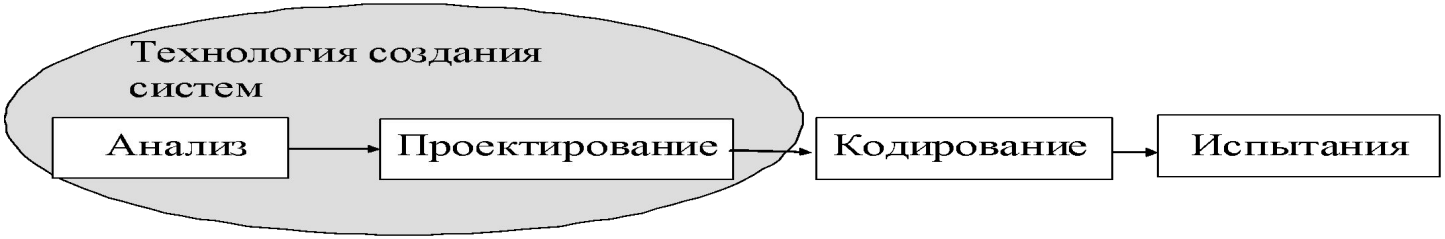
Жизненный цикл программного средства

Жизненный цикл программной системы представляет собой непрерывный процесс, начинающийся с момента принятия решения о ее создании и заканчивается в момент полного изъятия ее из эксплуатации.

Под ***моделью жизненного цикла*** понимается **схема**структура, определяющая последовательность выполнения **деятельностей, связанных с разработкой, эксплуатацией и сопровождением ПО** и их взаимосвязи (на протяжении жизненного цикла).

Стандарт ISO/IEC 12207 *не предопределяет конкретной модели жизненного цикла или метода разработки программного средства. Пользователи, применяющие настоящий стандарт, должны сами выбирать модель жизненного цикла применительно к своему программному проекту и распределять процессы, работы и задачи, выбранные из настоящего стандарта, на данной модели.*

Линейные модели



Линейная последовательная модель.

Каскадная модель (*однократный проход* , ***водопадная стратегия***, *waterfall model*) - **систематический подход к разработке программного обеспечения**, который начинается на системном уровне и проходит через анализ, проектирование, кодирование, испытания и сопровождение.

- 1970 -1985 гг

Документы, создаваемые при использовании каскадной модели

Вид деятельности	Выходные документы
Анализ требований	Исследование реализуемости Предварительные требования
Определение требований	Документ с требованиями
Спецификация системы	Спецификация функций План приемочных испытаний Предварительное руководство пользователя
Архитектурное проектирование	Спецификация архитектуры План испытаний системы
Проектирование интерфейса	Спецификация интерфейса План испытаний при комплексировании
Детальное проектирование	Спецификация проекта План испытаний программных единиц
Кодирование	Код программы
Испытание программных единиц	Отчет об испытаниях программных единиц
Испытания при комплексировании	Отчет об испытании при комплексировании Окончательное руководство пользователя
Испытание системы	Отчет об испытаниях системы
Приемочные испытания	Окончательный вариант системы и системная документация

Оценка каскадной модели

Преимущества:

- Достаточно легко при таком технологическом подходе вести планирование работ и формирование бюджета.
- она обеспечивает прямое определение контрольных точек на протяжении всего проекта
- вынуждает готовить огромную массу документации.

Проблемы:

- не годится для сложных проектов, когда требования могут неоднократно меняться; трудно приспосабливается к «неточной реальности», которая существует в начале проекта.
- Большинство грубых ошибок, которые не могут быть обнаружены до тех пор, пока программа не пройдет экспертизу, могут быть катастрофическими; все проблемы, связанные с ошибками или недочетами начальных этапов, выясняются только в процессе интеграции.
- Линейная природа классического жизненного цикла ведет к так называемым "блокирующим состояниям».

Из литературы известны также такие понятия как «**Каскадно-возвратный подход»,** претендующий на преодолени основного недостатка каскадн подхода - отсутствие гибкости Здесь разрешены возвраты к предыдущим стадиям и пересмотр или уточнение ран принятых решений.

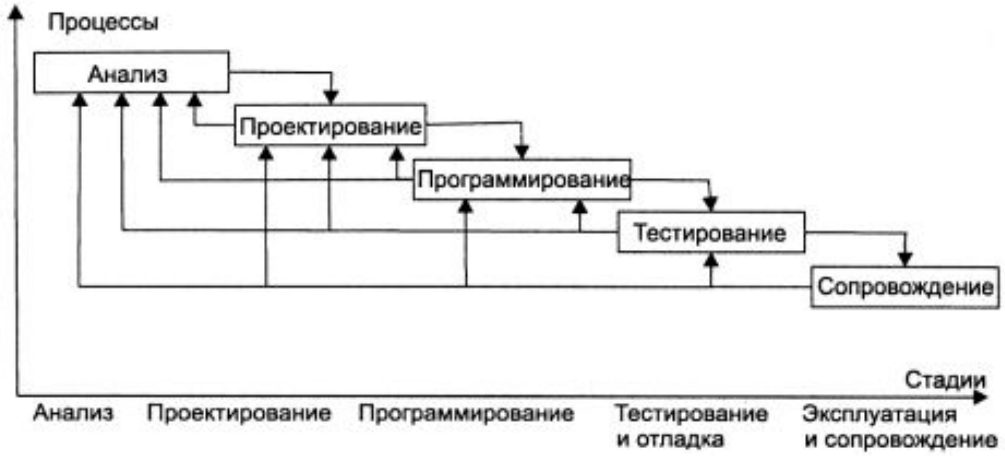
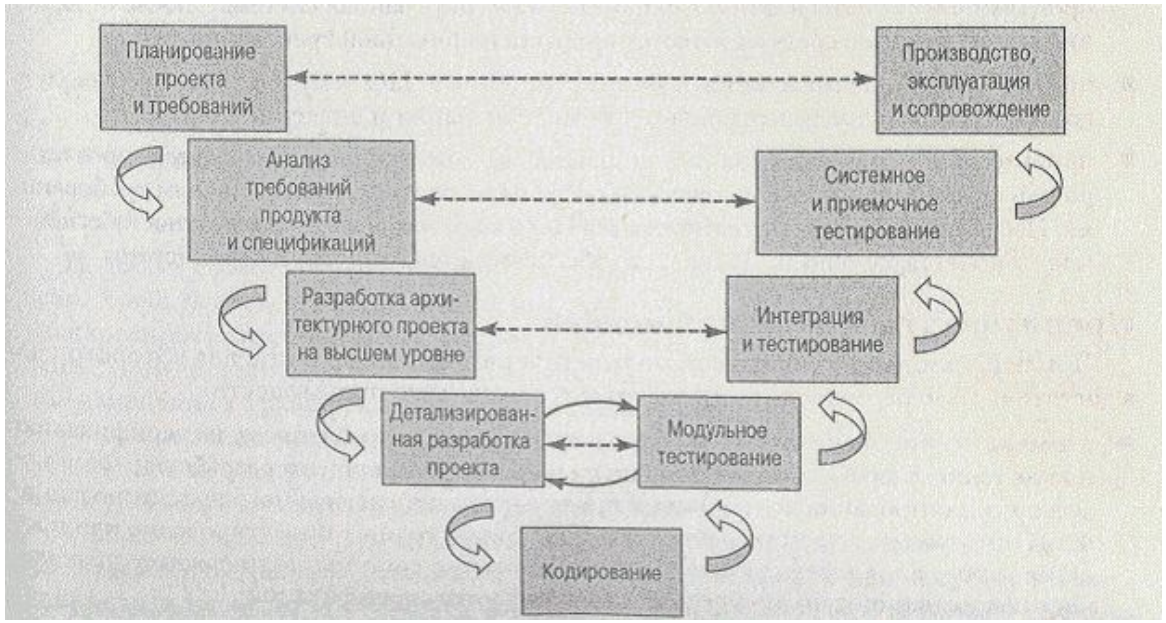


Рис. 3.6. Каскадно-возвратный технологический подход

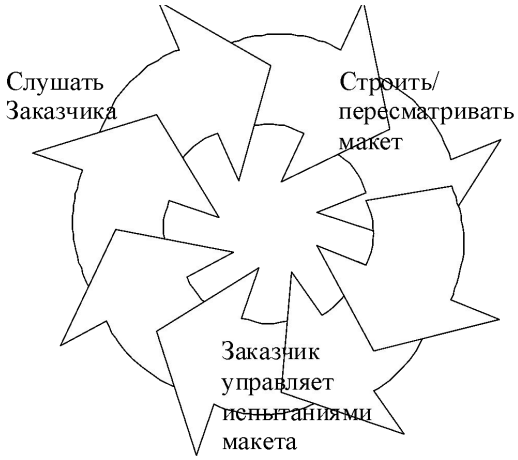
Каскадный подход с перекрывающимися процессами (waterfall with overlapping) предполагает наличие таких специализированных команд, позволяющих до определенной степени сократить передаваемую документацию. Следующий процесс начинается до завершения текущего:

V-образная модель жизненного цикла

Цель: **помочь** работающей над проектом команде **в планировании с обеспечением дальнейшей возможности тестирования системы**. В этой модели особое значение придается **действиям, направленным на верификацию и аттестацию продукта**.



Линейные модели



Парадигма прототипирования.

Модель с прототипированием

Цель: **поэтапное уточнение требований заказчика и получение законченной спецификации,** определяющей разрабатываемую систему.

Часто заказчик определяет ***множество общих целей*** для программного средства, но не идентифицирует детально входные данные, их обработку или требования к выходным данным. В других случаях разработчик может быть не уверен в эффективности алгоритмов, пригодности операционной системы или в форме, которую должно бы иметь человеко-машинное взаимодействие.

Важно - определение «правил игры» с самого начала, т.е. оба: заказчик и разработчик, должны согласиться, что **прототип строится, чтобы служить механизмом для определения требований.**

Оценка модели с прототипированием

Преимущества:

- возрастает вероятность полностью собранных требований;
- обеим сторонам нравится парадигма прототипирования. Пользователи получают ощущение реальной системы, а разработчики - возможность что-то создать немедленно.

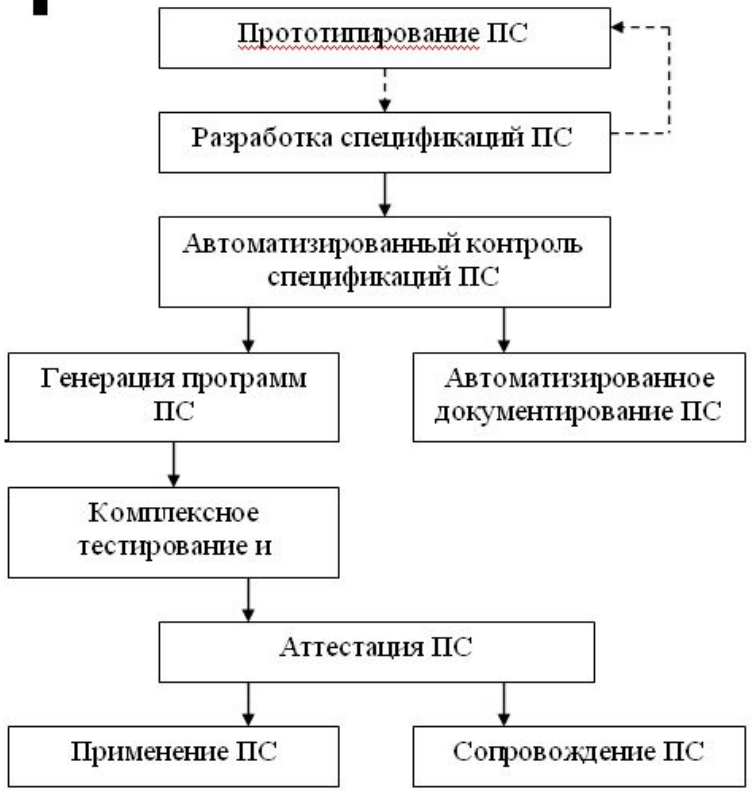
Проблемы:

- Заказчик видит, что появляется рабочая версия программного средства, не подозревая, что этот прототип собран наспех, и, настаивает, чтобы были осуществлены "несколько исправлений" для превращения прототипа в работающий продукт. Слишком часто менеджмент разработки программного обеспечения смягчается.
- Разработчик часто делает реализацию компромиссной для того, чтобы получить работающий прототип быстро. если не «выбрасывать» прототип, то менее чем идеальный выбор становится составной частью системы ;
- система часто получается плохо структурированной, т.к. постоянное внесение изменений приводит к упущениям в структуре ПО и становится все более сложным и затратным.

Линейные модели

Компьютерная ТРПО: используются программные инструменты для разработки формализованных спецификаций программ с последующей автоматической генерацией программ и документов.

Это разработка с использованием CASE-средств, **позволяющих разработчику «специфицировать желаемый результат**, а не действия, необходимые для достижения этого результата». Поддерживающее ПС транслирует эти спецификации результата в машинно-исполняемую программу.



Линейные модели

Методы четвертого поколения ("forth generation techniques" - 4GT) подразумевают *компьютерную ТРПО*, в которой желаемые **характеристики программного средства определяют на высоком уровне** и используют программные инструменты для автоматической генерации программ.

Разработчик определяет некоторые характеристики программного средства на уровне, который близок к естественному языку или использует ***нотацию, которая наделена важными функциями***, иначе говоря программист не пишет сам текст программы, а **с помощью некоторых *наглядных манипуляций указывает системе, какие задачи*** должны выполняться программой.

- В языки 4GL встраиваются *мощные* ***примитивы,*** **позволяющие одним оператором описать такую функциональность, для которой потребовалось бы множество (тысячи) строк кода** на языках младших поколений.
- Языки 4GL применяются в специализированных областях, где высоких результатов можно добиться, используя проблемно-ориент. языки, оперирующие конкретными понятиями узкой предметной области.

Оценка методов четвертого поколения

Преимущества модели:

- - позволяет увеличить производительность программистов (особенно для маленьких или средних приложений);
- - позволяет сократить число потенциальных ошибок; автоматическая генерация программ делает не нужной автономную отладку и тестирование программ;
- - появилась возможность автоматической генерации части документации и генерации тестов ;
- *сопровождение ПС* существенно упрощается, так как основные изменения делаются только в спецификациях;

Проблемы:

- использование 4GT для больших разработок ПО требует столько же или больше усилий для выполнения анализа, проектирования и испытаний.
- использование CASE-средств разработки приложений не очень распространено в сфере разработки промышленных приложений (из-за ограниченности возможностей большинства CASE-систем);
- любая система автоматизированного проектирования обладает своей спецификой и никогда не отражает всех требований того или иного пользователя на 100 %

Линейные модели

Формальные преобразования:

…для разработки систем, которые должны удовлетворить строгим требованиям **надежности,** безотказности и безопасности

Спецификация системных требований имеет ***вид*** ***детализированной формальной спецификации, записанной с помощью специальной математической нотации***.

Далее формальная спецификация путем последовательных формальных преобразований ***трансформируется в исполняемую программу***.

ФМ-проекты, реализованные в Великобритании (в вычислительной лаборатории Оксфордского университета) в 1990 и 1992 гг.

1) узел вычислений с плавающей точкой Inmos для суперкомпьютера Transputer T800

- формальная разработка микрокода с помощью алгебраических методов с компьютерной поддержкой позволила **сэкономить 12 месяцев на тестировании**

2) гигантская ***система обработки транзакций CICS (***совместный проект с IBM ***)***

- **с помощью ФМ разработано лишь около десятой части (**переписывание части спецификаций **на языке Z),**
- Эта десятая часть «вылилась» **в сотни тысяч строк кода и тысячи страниц спецификаций**, позволила ***снизить затраты на 9%*** по сравнению с разработкой традиционными методами (за счет уменьшения числа ошибок и повышения качества результирующего кода)

Оценка «формальных преобразований»

Преимущества модели:

- **гарантируют соответствие созданных систем их спецификациям**;
- базируются на таких методиках, как сокрытие информации, структурное программирование и поэтапное улучшение, которые полезны для создания качественного (прежде всего надежного) кода.
- ФМ-проекты могут обходиться столь же дешево, а то и дешевле, чем проекты, разработанные с применением обычных методов.

Проблемы:

- эти методы сложно использовать, они весьма ресурсоемки и требуют специальных знаний и опыта использования,
- функционирование большинства систем с трудом поддается описанию методом формальных спецификаций, большинство современных разработчиков считают их узкоспециализированными.
- не всегда дают существенного выигрыша в стоимости (несоответствие затрат и возврата от инвестиций),
- плохо подходят для проектирования пользовательского интерфейса, при котором важную роль играют неформальные соображения;
- Применять ФМ ко всем компонентам системы и излишне, и дорого.

Линейные модели

- **Сборочное программирование** - подход ,предполагающий, что ПС конструируется, главным образом, из компонентов, которые уже существуют. Идея:
- скорее «собирать» программу, используя множество существующих компонентов программного обеспечения, чем разрабатывать.
- В этом подходе начальный ***этап специфицирования требований*** **и** ***этап аттестации*** такие же, как в других моделях, а этапы между ними имеют такой смысл:
- ***Анализ компонентов*** – поиск компонентов, которые могли бы удовлетворять сформулированным требованиям, обычно невозможно точно сопоставить функции готовых компонентов и требуемые функции.
- ***Модификация требований*** – анализируются требования с учетом информации о компонентах, полученной на предыдущем этапе. Они модифицируются так, чтобы максимально использовать возможности отобранных компонентов.
- ***Проектирование*** – проектируется структура системы с учетом функциональных возможностей отобранных готовых программных компонентов. (компонент ПО должен «проектироваться повторноиспользуемым»).

Оценка сборочного программирования

Преимущества модели:

- Стоимость приобретения и объединения повторно-используемых компонентов будет почти всегда меньше, чем стоимость разработки эквивалентного программного обеспечения.
- С помощью концепций CBSE (component-based software engineering — и COTS (commercial off-the-shelf) разработка ПО превращается в инженерную дисциплину.

Проблемы:

- объединенные вместе высококачественные модули не обязательно превратятся в высококачественную систему. Комбинированная система может оказаться никуда негодной из-за некорректного способа объединения, либо из-за ошибочных представлений о поведении компонентов или о среде, в которую они помещаются.
- COTS-компоненты, которые обычно лицензируются в виде исполняемых модулей, могут породить неприятные побочные эффекты, неизвестные получателю лицензии. (они проявляются только при объединении компонентов, и их практически невозможно обнаружить при тестировании каждого модуля в отдельности).

Эволюционные модели программных процессов

- **Требования** к бизнесу и продуктам часто **изменяются**,
- **сжатые рыночные сроки** делают завершение всестороннего продукта невозможным.

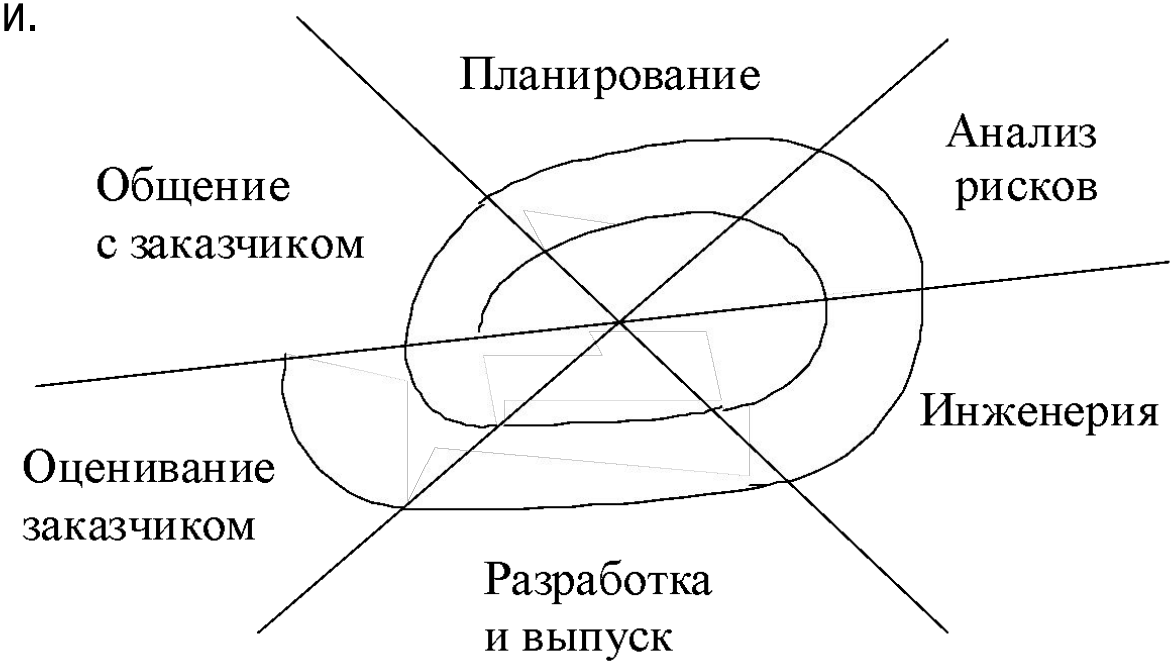
Инженеры ПО нуждаются в *эволюционной модели процессов* - циклическом повторении практически всех фаз.

Эволюционная стратегия: система строится в виде последовательности версий.

Спиральная модель процессов

Модель предусматривает циклическое **повторение практически всех этапов работ.**

При ранних итерациях уточняются спецификации продукта, «выпуск» может быть моделью на бумаге или прототипом. **При поздних** итерациях выпускаются все более увеличивающиеся по сложности версии сконструированной системы, добавляются новые возможности и функции.



Типичная спиральная модель.

Спиральная модель процессов

- ***Общение с заказчиком*** (customer communication) – задачи по установлению эффективного общения разработчика и заказчика с целью выявить требования к ПО.
- ***Планирование*** (planning) - задачи, необходимые для определения ресурсов, сроков и другой, связанной с проектом, информации.
- ***Анализ рисков*** (risk analysis) - задачи, необходимые для оценивания как технических рисков, так и рисков менеджмента.
- ***Инженерия*** (engineering) - задачи, необходимые для построения одного или более представлений приложения (спецификаций требований и проектов).
- ***Конструирование и выпуск*** (construction and release) - задачи, необходимые для построения, испытаний, **установки** и обеспечения поддержки пользователя (например, документация и обучение).
- ***Оценивание заказчиком*** (customer evaluation) - задачи, необходимые для получения обратной связи с заказчиком и основанные на оценивании представлений программного средства, созданных во время стадии инженерии и реализованных во время стадии установки.

Оценка спиральной модели процессов

Преимущества модели:

- Модель может быть приспособлена для применения на протяжении всей жизни программного средства.
- Дает реальную основу для управления рисками и сложностью в проекте. Позволяет выделять *работы, связанные с наиболее значительными рисками* в начальные итерации, *до того как в разработку вложены существенные средства*.

Проблемы:

- С точки зрения управления такой подход значительно сложнее; может быть трудна для убеждения заказчиков (в том, что эволюционный подход является управляемым).
- Во многих проектах, управляемых по спиральной модели, нет чётко определённого критерия окончания работы; «Неизвестная стоимость» затрудняет оценку стоимости (и доходности проекта).
- Для эффективности этого метода необходима высокая степень автоматизации процессов и документооборота.

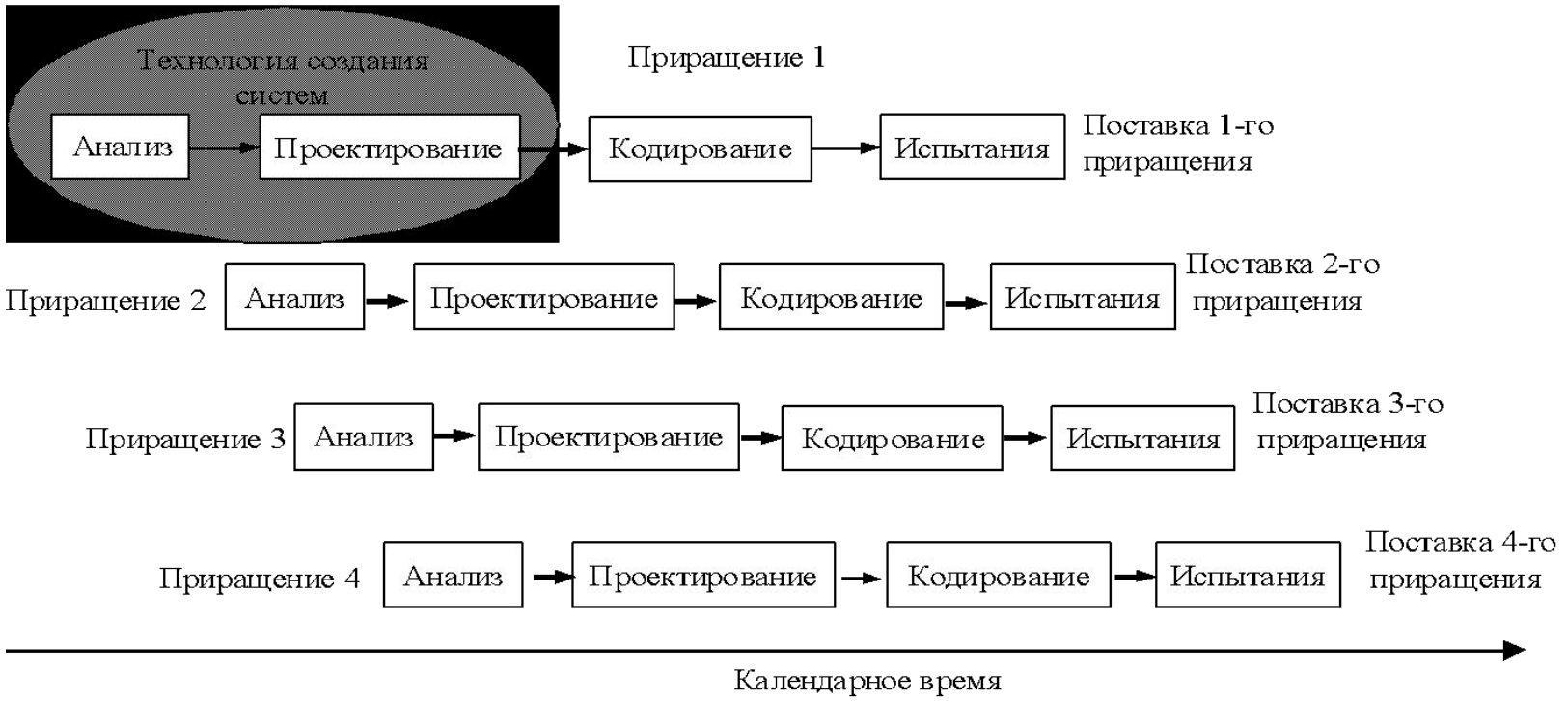
Модель разработки приращениями

...когда стесненные сроки могут помешать реализации всех требований ...

инкрементная стратегия: в начале процесса определяются все пользовательские и системные требования, оставшаяся часть конструирования выполняется в виде последовательности версий.

Изначально должны быть установлены *приоритеты требований.*

Первая версия реализует часть запланированных возможностей. *В конце каждой* итерации должна получаться работающая *версия продукта*, но с неполной функциональностью.



Модель разработки приращениями.

Оценка модели с приращениями

Преимущества:

- заказчику не надо ждать полного завершения системы, чтобы получить о ней представление;
- базовые функции подвергаются наиболее тщательной проверке, что снижает вероятность ошибок в особо важных частях системы.

Проблемы:

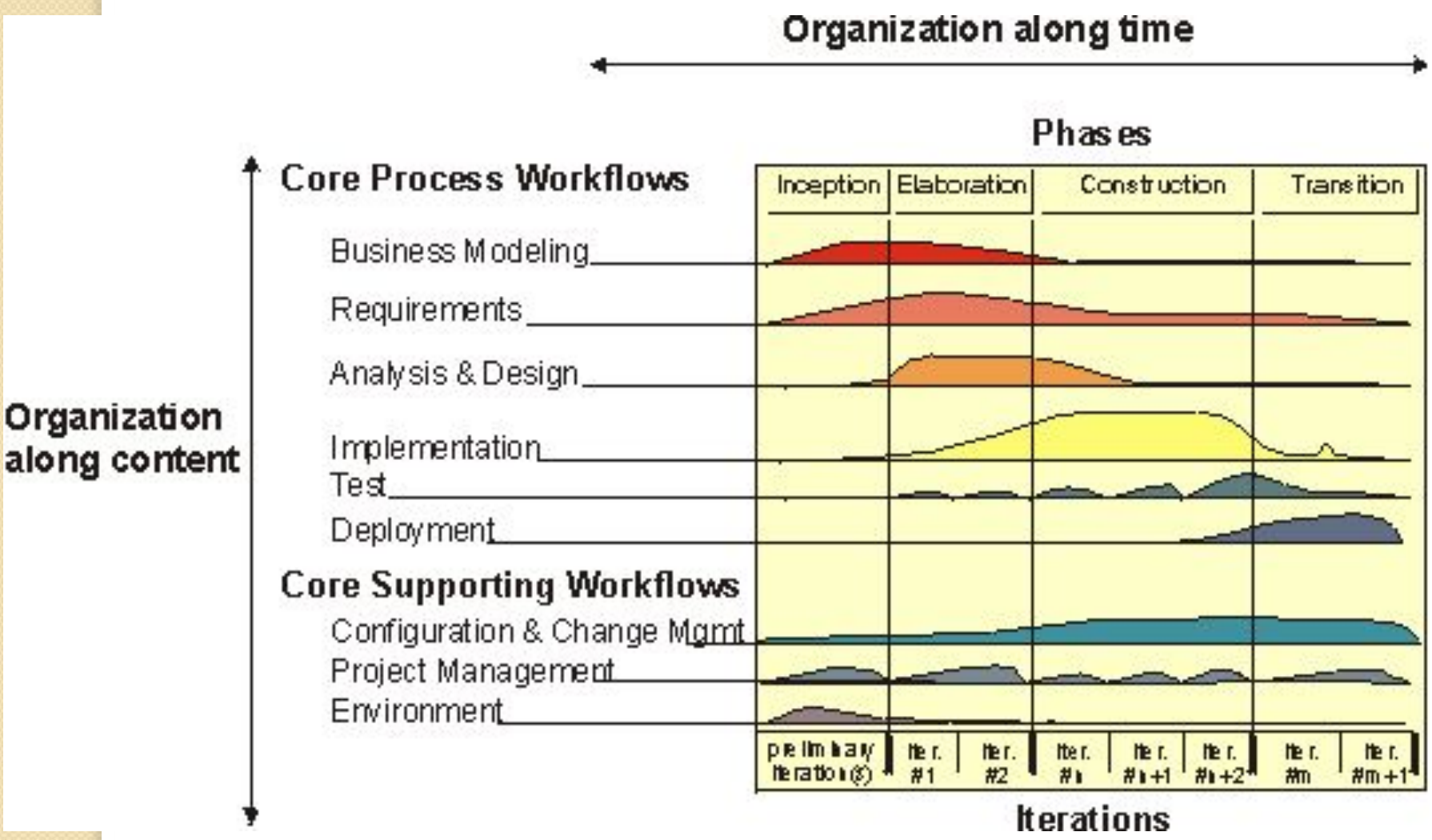
- На практике у заказчиков и пользователей иногда возникает ощущение нестабильности продукта, так как они не успевают уследить за слишком быстрыми изменениями в нём.
- При большом числе итераций разработка по этой модели нуждается в глубокой автоматизации всех процессов, иначе она становится неэффективной.


2
Жизненные циклы разработки
Итерационная модель
Итерация №1
Итерация №2
Итерация №3
R
D
C
T
Модель «Водопада»
Анализ требований
Проектирование
Кодирование и модульное тестирование
Тестирование подсистем
Системное тестирование
R Анализ требований
D Проектирование
C Кодирование и модульное тестирование
T Тестирование подсистем и системы

Рациональный унифицированный процесс

RUP (Rational Unified Process, RUP) - итеративный процесс, предполагающий разделение проекта на несколько «***мелких проектов***», которые выполняются последовательно (как бы несколько жизненных циклов разработки с применением модели "Водопада"), и каждая итерация (*фаза*) разработки четко определена набором целей, которые должны быть достигнуты:

- ***Исследование (Inception)***
- ***Уточнение\Проработка плана (Elaboration)***
- ***Построение (Construction)***
- ***Развертывание\«передача» (Transition)***


Organization along time
Phases
Inception
Elaboration
Construction
Transition
Core Process Workflows
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Core Supporting Workflows
Configuration & Change Mgmt
Project Management
Environment
Organization along content
Iterations