Программная инженерия

1 Раздел. Введение

2 Раздел. Жизненный цикл программного обеспечения

3 Раздел. Стандарты, регламентирующие ЖЦ ПО

Расписание:

11.01 Пн	14:30-18:25	Г-2082	Лекция
12.01 Вт	14:30-18:25	Г-2082	Лекция
13.01 Ср	14:30-18:25	Г-2023	Лабораторная работа
14.01 Чт	14:30-18:25	Г-2023	Лабораторная работа
21.01 Чт	18:30-21:40	Г-2023	Экзамен

1.Введение

Термин “Программная инженерия“ (software engineering)

- появился впервые в октябре 1968 г. на конференции подкомитета НАТО по науке и технике (г.Гармиш, Германия). На конференции рассматривались проблемы проектирования, разработки, распространения и поддержки ПО.
- Термин «**программная инженерия**» прозвучал как некоторая дисциплина, которую надо создавать и которой надо руководствоваться при решении перечисленных проблем
- **Программная инженерия** (или **технология промышленного программирования** – так называлась в России) возникла и развивалась под давлением роста стоимости создаваемого ПО.

Главная цель программной инженерии – сокращение стоимости и сроков разработки программ

“Программная инженерия " (software engineering).

Потребности

(в конце 70-х гг. XX в):

- Контролировать процесс разработки ИС,
- Прогнозировать и гарантировать стоимость разработки,
- Соблюдения сроков разработки
- Гарантировать качество результатов

Необходим переход от кустарных к индустриальным способам создания ИС.

Появилась совокупность инженерных методов и средств создания ИС, объединенных общим названием "программная инженерия " (software engineering).

Определение программной инженерии

IEEE Computer Society* определяет программную инженерию как

«применение систематизированного, дисциплинированного и оцениваемого по количественным параметрам подхода к разработке, функционированию и сопровождению программного обеспечения, то есть применение инженерного подхода к созданию ПО».

* - IEEE Computer Society - ведущая организация в области компьютерных наук, крупнейшее профессиональное сообщество, объединяющее более 100 тыс. специалистов, работающих в области компьютерных наук и технологий. Штаб-квартира находится в Вашингтоне;

Некоторые другие определения программной инженерии см. стр.9 литературы 4

Программная инженерия - дисциплина разработки ПО

Возникновение программной инженерии как дисциплины разработки ПО определено многими факторами, среди которых самыми важными являются следующие:

1. накопление в области создания ПО значительного объема разнородных знаний,
2. появление новых методов проектирования и разработки ПО;
3. совершенствование методов обнаружения ошибок в ПО;
4. эффективная организация работы коллективов разработчиков ПО.

Предпосылки возникновения
(в 4 литературе)

Повторное использование кода

модульное программирование

Рост сложности программ

структурное программирование

Модификация программ

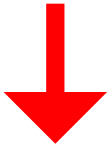
объектно-ориентированное программирование

Проблемы создания качественных компьютерных приложений

- ПО не использует потенциальные возможности аппаратуры;

- умение строить новые программы отстает от требований к программам;

- низкое качество разработки программ.



Ключ к решению этих проблем

- грамотная организация процесса создания ПО;
- реализация *технологических принципов промышленного конструирования* программных средств (ПС).

Организации разрабатывающие международные стандарты в области программной инженерии:

- ISO – International Organization for Standardization – Международная организация по стандартизации. Наиболее представительная и влиятельная организация, разрабатывающая стандарты почти во всех областях деятельности, в том числе и в IT.
- ACM – Association for Computing Machinery–Ассоциация по вычислительной технике. Всемирная научная и образовательная организация в области вычислительной техники. Известна также и разработкой образовательных стандартов
- SEI – Software Engineering Institute – Институт Программной Инженерии. Исследования в области программной инженерии с упором на разработку методов оценки и повышения качества ПО. Стандарты по качеству ПО и зрелости организаций, разрабатывающих ПО.
- PMI – Project Management Institute – Международный Институт Проектного Менеджмента (Управления Проектами). Некоммерческая организация, целью которой является продвижение, пропаганда, развитие проектного менеджмента в разных странах. PMI разрабатывает стандарты проектного менеджмента, занимается повышением квалификации специалистов.
- IEEE – Институт инженеров по электронике. Поддержка научных и практических разработок в области электроники и вычислительной техники. Большие вложения в разработку стандартов в этой области.

Наиболее известные стандарты программной инженерии

- ISO/IEC 12207 – Information Technology-Software Life Cycle Processes - Процессы жизненного цикла программных средств. Стандарт содержит определения основных понятий программной инженерии (в частности программного продукта и жизненного цикла программного продукта), структуры жизненного цикла как совокупности процессов, детальное описание процессов жизненного цикла. (см. раздел 3)

- SEICMM – Capability Maturity Model (for Software) - модель зрелости процессов разработки программного обеспечения. Стандарт отвечает на вопрос: «Какими признаками должна обладать профессиональная организация по разработке ПО?». Профессионализм организации определяется через зрелость процесса, применяемого этой организацией. Выделяются пять уровней зрелости процесса. (см. раздел 4)

Наиболее известные стандарты программной инженерии

- ISO/IEC 15504 – Software Process Assessment – Оценка и аттестация зрелости процессов создания и сопровождения ПО. Является развитием и уточнением ISO 12207 и SEICMM. Содержит расширенное по отношению ISO 12207 количество процессов жизненного цикла и 6 уровней зрелости процессов. Дается подробное описание схемы аттестации процессов, на основе результатов которой может быть выполнена оценка зрелости процессов и даны рекомендации по их усовершенствованию. (КР)

- PMBOK – Project Management Body of Knowledge – Свод знаний по управлению проектами. Содержит описания состава знаний по 9 разделам (областям знаний) управления проектами

Наиболее известные стандарты программной инженерии

- SWEBOK – Software Engineering Body of Knowledge – Свод знаний по программной инженерии – содержит описания состава знаний по 10 разделам (областям знаний) программной инженерии. (КР)

- ACM/IEEE CC 2001 – Computing Curricula 2001 – Академический образовательный стандарт в области компьютерных наук. Выделены 4 основных раздела компьютерных наук:
- Computer science,
- Computer engineering,
- Software engineering,
- Information systems,

по каждому из которых описаны области знаний соответствующего раздела, состав и планы рекомендуемых курсов

Что такое SWEBOK?

Guide to the Software Engineering Body of Knowledge (SWEBOK), IEEE 2004 Version- Руководство к Своду Знаний по Программной Инженерии [SWEBOK, 2004]

Назначение SWEBOK – объединение знаний по инженерии ПО. В этом ядре были систематизированы разнородные знания в области программирования, планирования и управления.

Для того чтобы задать границы программной инженерии как сферы профессиональной деятельности, SWEBOK вводит десять *областей знаний* (Knowledge Area, KA) программной инженерии.

SofWare Engineering (SWE – программная инженерия, или ранее – технология программирования)

Стр. 6 литература + статья **Знакомьтесь: SWEBOK(http://www.osp.ru/os/2006/07/3290839/)**

Цитата:

“Программная инженерия является развивающейся дисциплиной. Она не касается вопросов конкретизации применения тех или иных языков программирования.

Руководство к своду знаний SWEBOK включает базовое определение и описание областей знаний и является необходимым для понимания вопросов разработки ПО.

Одной из важнейших целей SWEBOK является именно определение тех аспектов деятельности, которые составляют суть профессии инженера-программиста.”

Программная инженерия. Проект SWEBOK
Казакова Ирина Анатольевна.
http://na-journal.ru/1-2012-tehnicheskie-nauki/44-programmnaja-inzhenerija-proekt-swebok

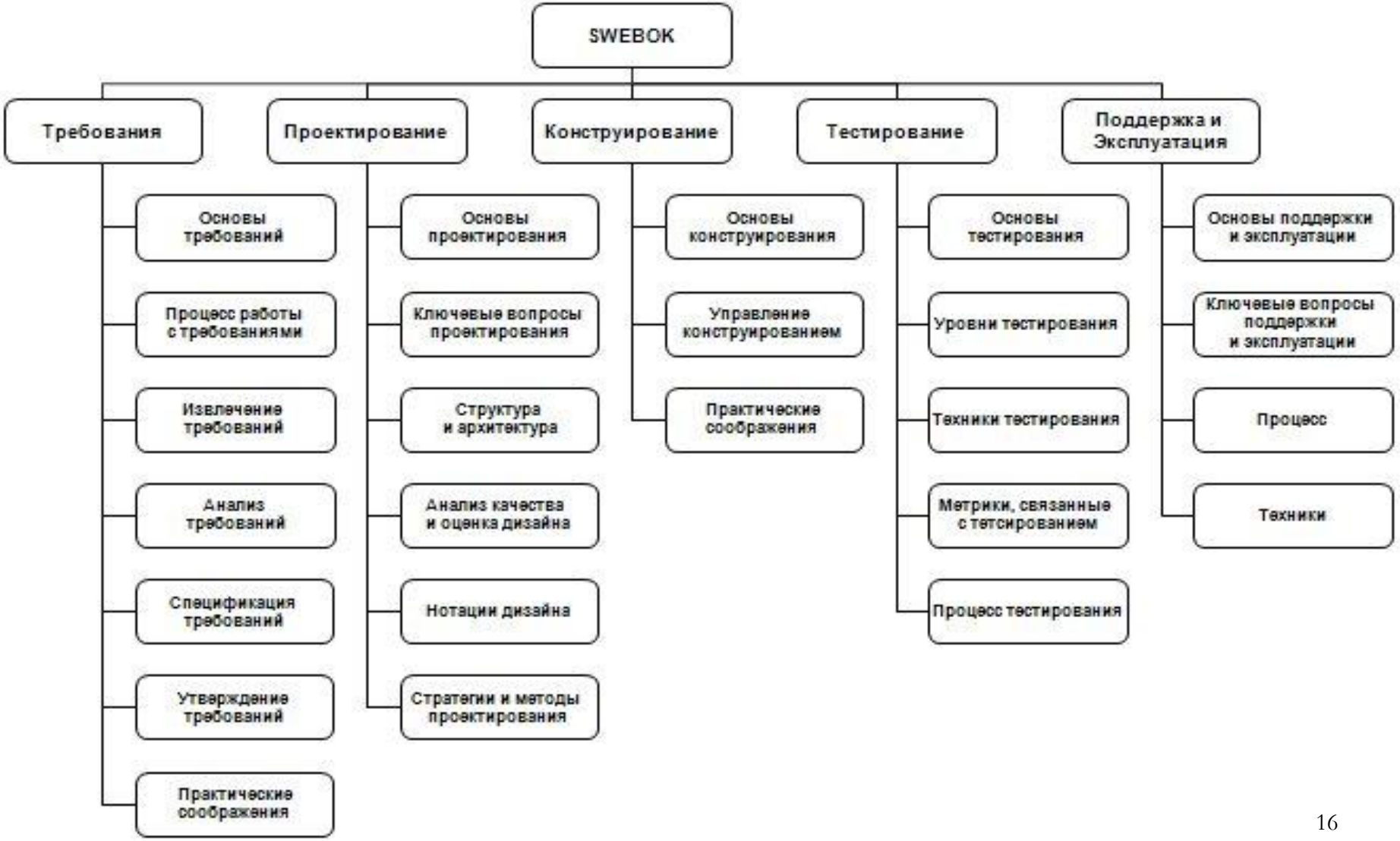
10 областей знаний (Knowledge Area)

которые соответствуют процессам проектирования ПО и методам их поддержки, а именно:

1. Software requirements –требования к ПО.
2. Software design – проектирование ПО.
3. Software construction – конструирование ПО.
4. Software testing - тестирование ПО.
5. Software maintenance – сопровождение ПО.
6. Software configuration management –управление конфигурацией.
7. Software engineering management – управление в программной инженерии.
8. Software engineering process – процессы программной инженерии.
9. Software engineering tools and methods – инструменты и методы программной инженерии.
10. Software quality – качество ПО.

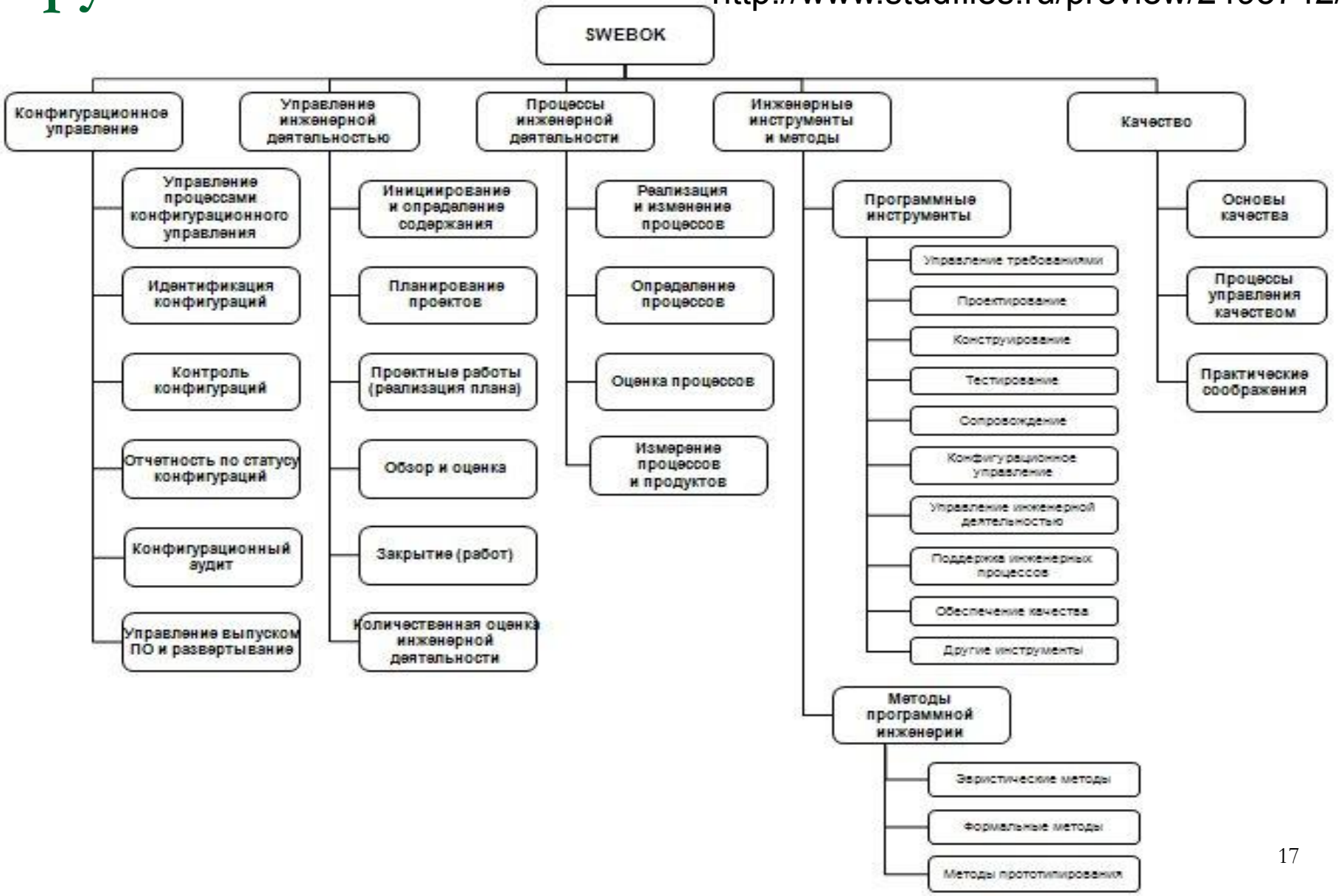
Первые 5 областей знаний SWEBOK на русском языке

http://www.studfiles.ru/preview/2495742/



Вторые 5 областей знаний SWEBOK на русском языке

http://www.studfiles.ru/preview/2495742/



SWEBOK также включает обзор смежных дисциплин:

связь с которыми представлена как фундаментальная, важная и обоснованная для программной инженерии:

1. Computer engineering – разработка компьютеров.
2. Computer science – информатика.
3. Management – общий менеджмент.
4. Mathematics – математика.
5. Project management – управление проектами.
6. Quality management – управление качеством.
7. Systems engineering – системное проектирование.

Примерные вопросы по SWEBOK

SWEBOK. Область знаний – требования

- Что такое требование. Какие виды требований различают. Какие характеристики требований.
- Какие разделы включены в эту область знаний SWEBOK
- Методы выявления/извлечения/анализа/проверки требований

SWEBOK. Область знаний – качество ПО

- Что такое качество ПО.
- Какие разделы включены в эту область знаний SWEBOK
- Процессы управления качеством ПО
- Понятия верификации, аудита, аттестации и пр.
- Какие разделы включены в эту область знаний SWEBOK

Технология конструирования программного обеспечения (ТКПО) ТКПО

— система инженерных принципов для создания экономичного ПО, которое надежно и эффективно работает в реальных компьютерах

Различают методы, средства и процедуры ТКПО

Методы обеспечивают:

- планирование и оценка проекта;
- анализ системных и программных требований;
- проектирование алгоритмов, структур данных и программных структур;
- кодирование;
- тестирование;
- сопровождение.

Процедуры

- соединяют методы и средства в непрерывную технологическую цепь разработки.

Процедуры определяют:

- порядок применения методов и средств;
- формирование отчетов в соответствии с требованиями;
- контроль качества и координирование изменений;
- формирование “вех” (промежуточных этапов) для оценки прогресса.

Средства (утилиты) ТКПО

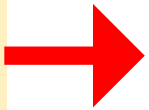
обеспечивают автоматизированную или автоматическую реализацию методов с помощью CASE - систем.

Процесс конструирования программного обеспечения состоит из последовательности шагов, использующих **методы, утилиты и процедуры**. Эти последовательности шагов часто называют **парадигмами ТКПО**.

“Главная идея” software engineering

фундаментальная идея программной инженерии: **“проектирование ПО ИС является формальным процессом, который можно изучать и совершенствовать.”**

Освоение и правильное использование методов и средств создания ПО позволят повысить качество ИС, обеспечить управляемость процесса проектирования ИС и увеличить срок ее жизни.



Появление программно-технологических средств специального класса – CASE-средств, реализующих CASE-технологию создания и сопровождения ИС.

Термин CASE

Computer Aided Software Engineering (англ.) - дословный перевод: разработка программного обеспечения информационных систем при поддержке (с помощью) компьютера.

Первоначальное значение термина CASE, ограничено вопросами автоматизации разработки только лишь программное обеспечение (ПО). Однако:

«Термин “CASE” в настоящее время используется в широком смысле и не ограничивается вопросами автоматизации разработки только лишь ПО, но и охватывает процесс разработки сложных информационных систем в целом».

Формулировка официального сайта Санкт-Петербургского информационно-аналитического центра (СПбИАЦ)

Предпосылки появления средств автоматизации:

Ручная разработка обычно порождала следующие проблемы:

- *неадекватная спецификация требований;*
- *неспособность обнаруживать ошибки в проектных решениях;*
- *низкое качество документации, снижающее эксплуатационные качества;*
- *затяжной цикл и неудовлетворительные результаты тестирования.*

CASE:

CASE-инструменты (CASE-средства)

– инструментарий для системных аналитиков, разработчиков и программистов, заменяющий им бумагу и карандаш на компьютер для автоматизации процесса проектирования и разработки ПО.

CASE-технологии

совокупность методологий анализа, проектирования, разработки и сопровождения сложных систем ПО, поддержанная комплексом взаимоувязанных средств автоматизации.

CASE-индустрия объединила сотни фирм и компаний различной ориентации

Понятие CASE-технология

CASE-технология представляет собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных систем и поддерживается комплексом взаимоувязанных средств автоматизации. CASE-технология - это инструментарий для системных аналитиков, разработчиков и программистов, заменяющий бумагу и карандаш компьютером, автоматизируя процесс проектирования и разработки ПО.

CASE-технологии =
методология разработки ПО + CASE-средства

Свойства хорошей программы

Программа прежде всего должна удовлетворять требованиям заказчика – это **функциональные требования.**

Есть также **нефункциональные требования**:

- Сопровождаемость
- Надежность (отказоустойчивость, безопасность, защищенность)
- Эффективность
- Удобство использования

Стр. 16-17 - 4 литература

Таблица. Основные показатели качественного ПО(2 лит-ра стр.14)

Показатель	Описание
Удобство сопровождения	ПО должно быть таким, чтобы существовала возможность его усовершенствования в ответ на измененные требования заказчика или пользователя. Это определяющий показатель, поскольку любое ПО неминуемо подвергается модернизации вследствие изменений, происходящих в реальном мире
Надежность	Определяется рядом характеристик, таких как безотказность, защищенность и безопасность. Надежность ПО означает, что возможные сбои в работе системы не приведут к физическому или экономическому ущербу
Эффективность	Работа ПО не должна приводить к расточительному расходованию таких системных ресурсов, как память или время занятости процессора. Поэтому эффективность ПО описывается следующими характеристиками: скорость выполнения, используемое процессорное время, объем требуемой памяти и т.п.
Удобство в использовании	ПО должно быть удобным в эксплуатации и не требовать чрезмерного напряжения усилий пользователя того уровня, на которого оно рассчитано. Это означает, что программная система должна обладать соответствующим пользовательским интерфейсом и необходимой документацией

Вопросы и ответы об инженерии ПО (из лит-ры 2)

Вопрос	Ответ
Что такое программное обеспечение (ПО)?	Это компьютерные программы и соответствующая документация. Программные продукты разрабатываются или по частному заказу, или для продажи на рынке программных продуктов
Что такое инженерия программного обеспечения?	Это инженерная дисциплина, охватывающая все аспекты разработки программного обеспечения
В чем различие между инженерией программного обеспечения и компьютерной наукой?	Компьютерная наука – это теоретическая дисциплина, охватывающая все стороны вычислительных систем, включая аппаратные средства и программное обеспечение; инженерия программного обеспечения – практическая дисциплина создания и сопровождения программных систем
В чем различие между инженерией программного обеспечения и системотехникой?	Системотехника охватывает все аспекты разработки вычислительных систем (включая создание аппаратных средств и ПО) и соответствующие технологические процессы. Технологии инженерии программного обеспечения являются частью этих процессов
Что такое технологический процесс создания ПО?	Это совокупность процессов, ведущих к созданию или развитию программного обеспечения
Что такое модель технологического процесса создания ПО?	Формализованное упрощенное представление технологического процесса создания ПО

Вопросы и ответы об инженерии ПО(из лит-ры 2)

Вопрос	Ответ
Какова структура затрат на создание ПО?	Примерно 60% от общих затрат на создание ПО занимают затраты непосредственно на разработку ПО и 40% – на его тестирование и отладку. Для программных продуктов, разрабатываемых по заказу, стоимость тестирования и отладки часто превышает стоимость разработки продукта
Что такое методы инженерии программного обеспечения?	Это структурные решения, предназначенные для разработки ПО и включающие системные модели, формализованные нотацию и правила проектирования, а также способы управления процессом создания ПО
Что такое CASE (Computer-Aided Software Engineering – автоматизированное проектирование и создание ПО)?	Это программные системы, предназначенные для автоматизации процесса создания ПО. CASE-средства часто используются в качестве основы методов инженерии программного обеспечения
Каковы признаки качественного ПО?	Программные продукты должны удовлетворять требованиям функциональности и эффективности (с точки зрения пользователя), а также быть надежными, удобными в эксплуатации и иметь возможности для модернизации
Какие основные проблемы стоят перед специалистами по программному обеспечению?	Проблема наследования ранее созданного ПО, проблема все возрастающей разнородности программных систем и проблема, порожденная требованием уменьшения времени на создание ПО

2. Жизненный цикл программного обеспечения(ЖЦ ПО)

Важнейшей и старейшей парадигмой процесса разработки ПО является жизненный цикл(ЖЦ)

Впервые о необходимости рассматривать разработку ПО с позиции его ЖЦ “заговорили” еще в 1968 г. - Классический ЖЦ - каскадная или водопадная модель (автор Уинстон Ройс, 1970)

Модели процесса создания ПО

- Центральный объект изучения программной инженерии - **процесс создания ПО** – множество различных видов деятельности, методов, методик и шагов, используемых для разработки и эволюции ПО и связанных с ним продуктов (проектных планов, документации, программного кода, тестов, пользовательской документации и пр.).
- Не существует **универсального процесса** разработки ПО – набора методик, правил и предписаний, подходящих для ПО любого вида, для любых компаний, для команд любой национальности.
- Процесс создания программного обеспечения не является однородным. Тот или иной метод разработки ПО, как правило, определяет некоторую динамику развертывания тех или иных видов деятельности, то есть, определяет ***модель процесса* (process model).**

Понятие ЖЦ

Жизненный цикл изделия(продукта) – совокупность всех действий, которые необходимо выполнить на протяжении всей «жизни» изделия. Смысл ЖЦ состоит во взаимосвязанности всех этих действий.

ЖЦ ПО – это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации. Таким образом, "**жизненный цикл ПО**" является более широким понятием, чем **модель процесса создания ПО**. Вместе с тем каскадную модель можно рассматривать как одну из моделей жизненного цикла ПО.

Определение из лит-ры 2

Модели процесса создания ПО

Водопадная (каскадная, последовательная) модель

предложена в 1970 г. Уинстоном Ройсом

Итерационная (инкрементная, эволюционная) модель

Спиральная модель

разработана в середине 1980-х Барри Боэмом

Один из вар-тов эволюционной модели

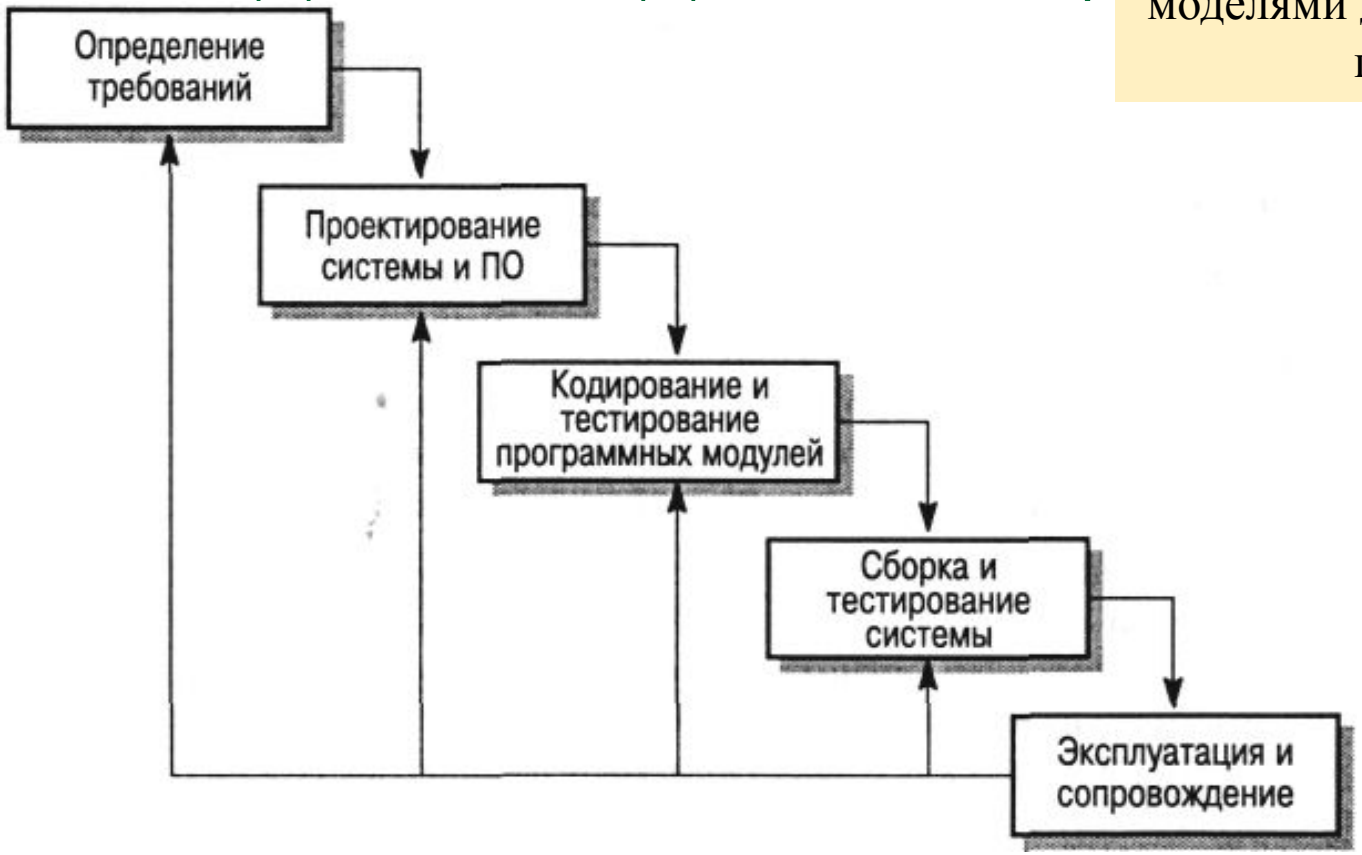
Модель - хорошая абстракция различных методов разработки ПО, позволяющая лаконично, сжато и информативно их представить.
Сама идея модели процесса является одной из самых ранних в программной инженерии, когда считалось, что удачная модель – самое главное, что способствует успеху разработки. Позднее пришло осознание, что существует множество других аспектов (принципы управления и разработки, структуру команды и т.д.), которые должны быть определены согласовано друг с другом.

из лит-ры 2

Водопадная (каскадная, последовательная) модель

Каскадная модель ЖЦ

Это первая модель процесса создания ПО, порожденная моделями других инженерных процессов



Каскадная модель – предполагает переход на следующую стадию после полного завершения работ предыдущей:

результат каждого этапа должен утверждаться документально (это как бы сигнал об окончании этапа). Тогда следующий этап не может начаться до завершения предыдущего. Однако на практике этапы могут перекрываться с постоянным перетеканием информации от одного этапа к другому.

Основные принципиальные этапы водопадной модели отражают все базовые виды деятельности, необходимые для создания ПО:

1. *Анализ и формирование требований.* Путем консультаций с заказчиком ПО определяются функциональные возможности, ограничения и цели создаваемой программной системы.
2. *Проектирование системы и программного обеспечения.* Процесс проектирования системы разбивает системные требования на требования, предъявляемые к аппаратным средствам, и требования к программному обеспечению системы. Разрабатывается общая архитектура системы. Проектирование ПО предполагает определение и описание основных программных компонентов и их взаимосвязей.
3. *Кодирование и тестирование программных модулей.* На этой стадии архитектура ПО реализуется в виде множества программ или программных модулей. Тестирование каждого модуля включает проверку его соответствия требованиям к данному модулю.

из лит-ры 2

4. *Сборка и тестирование системы.* Отдельные программы и программные модули интегрируются и тестируются в виде целостной системы. Проверяется, соответствует ли система своей спецификации.

5. *Эксплуатация и сопровождение системы.* Обычно (хотя и не всегда) это самая длительная фаза жизненного цикла ПО. Система инсталлируется, и начинается период ее эксплуатации. Сопровождение системы включает исправление ошибок, которые не были обнаружены на более ранних этапах жизненного цикла, совершенствование системных компонентов и "подгонку" функциональных возможностей системы к новым требованиям.

Сопровождение — это внесение изменений в эксплуатируемое ПО. Цели изменений:

- исправление ошибок;
- адаптация к изменениям внешней для ПО среды;
- усовершенствование ПО по требованиям заказчика.

Сопровождение ПО состоит в повторном применении каждого из предшествующих шагов (этапов) ЖЦ к существующей программе, но не в разработке новой программы.

из лит-ры 2

Достоинства классического жизненного цикла:

дает план и временной график по всем этапам проекта, упорядочивает ход конструирования; выполняемые в логичной последовательности этапы работ достаточно очевидным образом позволяли планировать сроки завершения всех работ и соответствующие затраты

Недостатки классического жизненного цикла:

1) реальные проекты часто требуют отклонения от стандартной последовательности шагов;

2) цикл основан на точной формулировке исходных требований к ПО (реально в начале проекта требования заказчика определены лишь частично);

3) результаты проекта доступны заказчику только в конце работы.

Поэтому каскадная модель применяется тогда, когда требования формализованы достаточно четко и корректно. Вместе с тем каскадная модель хорошо отражает практику создания ПО. Технологии создания ПО, основанные на данной модели, используются повсеместно, в частности **для разработки систем, входящих в состав больших инженерных проектов.**

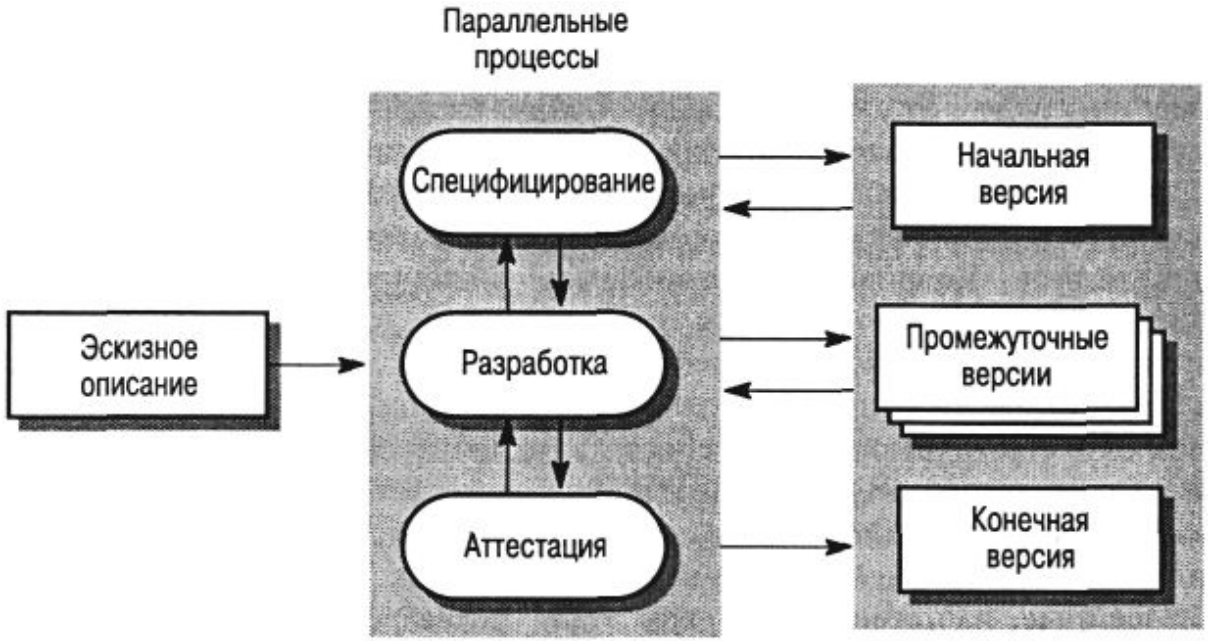
из лит-ры 2

Итерационная (инкрементная, эволюционная) модель

Эволюционная модель ЖЦ

Эта модель основана на следующей идее: разрабатывается первоначальная версия программного продукта, которая передается на испытание пользователям, затем она дорабатывается с учетом мнения пользователей, получается промежуточная версия продукта, которая также проходит "испытание пользователем", снова дорабатывается и так несколько раз, пока не будет получен необходимый программный продукт.

Отличительной чертой данной модели является то, что процессы спецификирования, разработки и аттестации ПО выполняются параллельно при постоянном обмене информацией между ними.



из лит-ры 2

Подходы к реализации эволюционного метода разработки

Подход пробных разработок.

Здесь большую роль играет постоянная работа с заказчиком (или пользователями) для того, чтобы определить полную систему требований к ПО, необходимую для разработки конечной версии продукта. В рамках этого подхода вначале разрабатываются те части системы, которые очевидны или хорошо специфицированы. Система эволюционирует (дорабатывается) путем добавления новых средств по мере их предложения заказчиком.

Прототипирование.

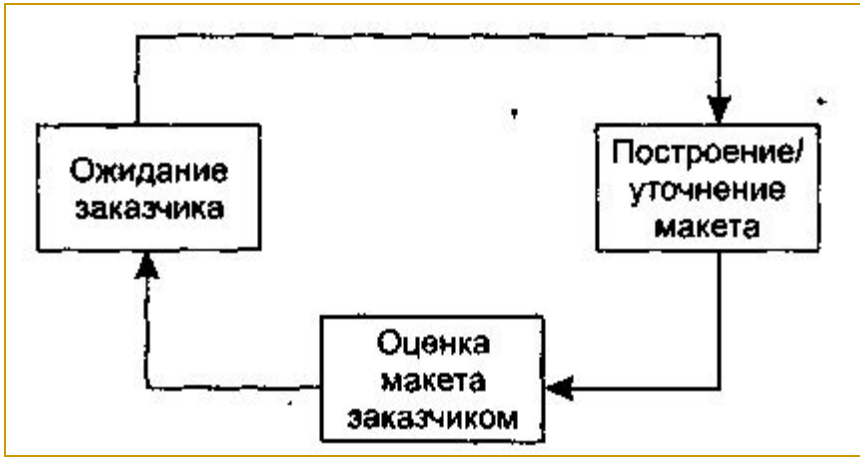
Здесь целью процесса эволюционной разработки ПО является поэтапное уточнение требований заказчика и, следовательно, получение законченной спецификации, определяющей разрабатываемую систему. Прототип обычно строится для экспериментирования с той частью требований заказчика, которые сформированы нечетко или с внутренними противоречиями.

из лит-ры 2

Прототипирование (макетирование)

— это процесс создания модели требуемого программного продукта. Модель может принимать одну из трех форм:

- бумажный макет или макет на основе ПК;
- работающий макет;
- существующая программа.



Основная цель макетирования — снять неопределенности в требованиях заказчика.

Достоинство макетирования:

- обеспечивает определение полных требований к ПО.

Недостатки макетирования:

- заказчик может принять макет за продукт;
- разработчик может принять макет за продукт.

из лит-ры 1

Последовательность действий при макетировании

из лит-ры 1

Примерные вопросы по Прототипированию

- Понятие прототипирования
- Технологии и методы быстрого прототипирования
- Применение для различных проектов
- Достоинства и недостатки

Инкрементная модель

Каждая линейная последовательность здесь вырабатывает поставляемый инкремент ПО.
В отличие от макетирования, инкрементная модель обеспечивает на каждом инкременте работающий продукт.

Современная реализация *инкрементного подхода* — экстремальное программирование (XP) (Кент Бек, 1999). XP ориентировано на очень малые приращения функциональности.

из лит-ры 1

Выводы по Эволюционной модели

Достоинства: часто более эффективен, чем подход, построенный на основе каскадной модели, особенно если требования заказчика могут меняться в процессе разработки системы. спецификация может разрабатываться постепенно, по мере того как заказчик (или пользователи) осознает и сформулирует те задачи, которые должно решать ПО.

Недостатки:

1. Многие этапы процесса создания ПО не документированы. Менеджерам проекта создания ПО необходимо регулярно документально отслеживать выполнение работ. Но если система разрабатывается быстро, то экономически не выгодно документировать каждую вер. системы.

2. Система часто получается плохо структурированной.

Постоянные изменения в требованиях приводят к ошибкам и упущениям в структуре ПО. Со временем внесение изменений в систему становится все более сложным и затратным.

3. Часто требуются специальные средства и технологии разработки ПО.

Это вызвано необходимостью быстрой разработки версий программного продукта. Но, с другой стороны, это может привести к несовместимости некоторых применяемых средств и технологий, что, в свою очередь, требует наличия в команде разработчиков специалистов высокого уровня.

из лит-ры 2

Спиральная модель

Спиральная модель является не альтернативой эволюционной модели, а специально проработанным её вариантом.

По Википедии

Спиральная модель ЖЦ

Спиральная модель (*spiral model*) была разработана в середине 1980-х годов Барри Боэмом. Она основана на классическом цикле Деминга PDCA (plan-do-check-act). При использовании этой модели ПО создается в несколько итераций (витков спирали) методом прототипирования.

Спиральная модель: *1* - начальный сбор требований и планирование проекта; *2* — та же работа, но на основе рекомендаций заказчика; *3* — анализ риска на основе начальных требований; *4* — анализ риска на основе реакции заказчика; *5* — переход к комплексной системе; *6* — начальный макет системы; *7* — следующий уровень макета; *8* — сконструированная система; *9* — оценивание заказчиком

Спиральная модель – делает упор на начальные этапы ЖЦ: анализ и проектирование. Каждый виток спирали соответствует поэтапной модели создания фрагмента или версии системы, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы следующего витка спирали.

Спиральная модель ЖЦ

Каждый виток имеет следующую структуру (секторы):

- *Планирование* - определение целей, ограничений и альтернатив проекта;
- *Анализ риска* - оценка альтернатив, оценка и разрешение рисков; возможно использование прототипирования (в том числе создание серии прототипов), симуляция системы, визуальное моделирование и анализ спецификаций; фокусировка на самых рисковых частях проекта;
- *Конструирование* - разработка и тестирование продукта следующего уровня – здесь возможна водопадная модель или использование иных моделей и методов разработки ПО;
- *Оценивание* - планирование следующих итераций – анализируются результаты, планы и ресурсы на последующую разработку, принимается (или не принимается) решение о новом витке; анализируется, имеет ли смысл продолжать разрабатывать систему или нет; разработку можно и приостановить, например, из-за сбоев в финансировании; спиральная модель позволяет сделать это корректно.

Спиральная модель ЖЦ

На каждой итерации оцениваются:

- риск превышения сроков и стоимости проекта;
- необходимость выполнения ещё одной итерации;
- степень полноты и точности понимания требований к системе;
- целесообразность прекращения проекта.

Отличительной особенностью спиральной модели является специальное внимание, уделяемое **рискам,** влияющим на организацию жизненного цикла, и **контрольным точкам**.

Спиральная модель является не альтернативой эволюционной модели, а специально проработанным её вариантом.

Боэм формулирует 10 наиболее распространённых (по приоритетам) рисков:

- Дефицит специалистов.
- Нереалистичные сроки и бюджет.
- Реализация несоответствующей функциональности.
- Разработка неправильного пользовательского интерфейса.
- Перфекционизм (ненужная оптимизация и оттачивание деталей).
- Непрекращающийся поток изменений.
- Нехватка информации о внешних компонентах, определяющих окружение системы или вовлеченных в интеграцию.
- Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами.
- Недостаточная производительность получаемой системы.
- Разрыв в квалификации специалистов разных областей.

Набор контрольных точек:

- Concept of Operations (COO) — концепция (использования) системы;
- Life Cycle Objectives (LCO) — цели и содержание жизненного цикла;
- Life Cycle Architecture (LCA) — архитектура жизненного цикла; здесь же возможно говорить о готовности концептуальной архитектуры целевой программной системы;
- Initial Operational Capability (IOC) — первая версия создаваемого продукта, пригодная для опытной эксплуатации;
- Final Operational Capability (FOC) —— готовый продукт, развернутый (установленный и настроенный) для реальной эксплуатации.

Спиральная модель ЖЦ

Достоинства:

- наиболее реально отображает разработку ПО;
- позволяет явно учитывать риск на каждом витке эволюции разработки;
- включает шаг системного подхода в итерационную структуру разработки;
- использует моделирование для уменьшения риска и совершенствования программного изделия.

Недостатки:

- новизна (отсутствует достаточная статистика эффективности модели);
- повышенные требования к заказчику;
- трудности контроля и управления временем разработки.

Стратегии конструирования ПО

Существуют 3 стратегии конструирования ПО:

- ***однократный проход* (водопадная стратегия)** — линейная последовательность этапов конструирования;
- ***инкрементная стратегия.*** В начале процесса определяются все пользовательские и системные требования, оставшаяся часть конструирования выполняется в виде последовательности версий. Первая версия реализует часть запланированных возможностей, следующая версия реализует дополнительные возможности и т. д., пока не будет получена полная система;
- ***эволюционная стратегия***. Система также строится в виде последовательности версий, но в начале процесса определены не все требования. Требования уточняются в результате разработки версий.

Характеристики стратегий конструирования ПО в соответствии с требованиями стандарта **IEEE/EIA12207.2** приведены в таб. ниже

из лит-ры 1

Характеристики стратегий конструирования ПО в соответствии с требованиями стандарта IEEE/EIA 12207.2

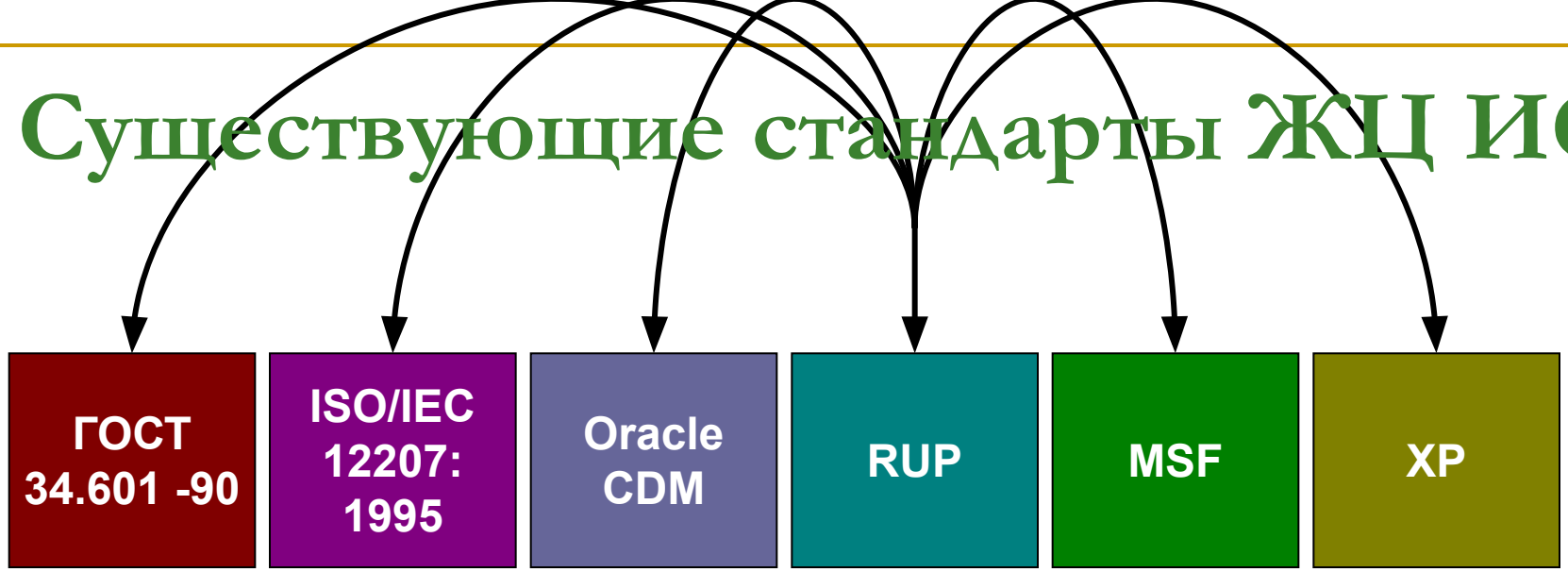
Стратегия конструирования	В начале процесса определены все требования?	Множество циклов конструирования?	Промежуточное ПО распространяется?
Однократный подход	Да	Нет	Нет
Инкрементная (запланированное улучшение продукта)	Да	Да	Может быть
Эволюционная	Нет	Да	Да

3.Стандарты ЖЦ ИС

Существует целый ряд стандартов, регламентирующих ЖЦ ПО, а в некоторых случаях и процессы разработки.

Стандарты используют различные модели ЖЦ ПО

Существующие стандарты ЖЦ ИС



ГОСТ 34.601-90 - распространяется на автоматизированные системы и устанавливает стадии и этапы их создания. Кроме того, в стандарте содержится описание содержания работ на каждом этапе. Стадии и этапы работы, закрепленные в стандарте, в большей степени соответствуют *каскадной модели* жизненного цикла

Процессы жизненного цикла ISO/IEC 12207 (принят в 1995)

В 2000 г. он был принят как ГОСТ (ИСО/МЭК 12207

- Процессы жизненного цикла программных средств

Для поддержки практического применения ISO/IEC 12207 разработан ряд технологических документов:

Руководство для ISO/IEC 12207 (ISO/IEC TR 15271:1998 Information technology - Guide for ISO/IEC 12207)

Руководство по применению ISO/IEC 12207 к управлению проектами (ISO/IEC TR 16326:1999 Software engineering - Guide for the application of ISO/IEC 12207 to project management)

ISO/IEC 12207

В основе практически всех современных промышленных технологий создания ПС лежит международный стандарт ISO/IEC 12207 «Системная и программная инженерия. Процессы жизненного цикла программных средств.»

Стандарт ISO12207 равносильно ориентирован на организацию действий каждой из двух сторон:

- поставщик (разработчик) и
- покупатель (пользователь); может быть в равной степени применен, когда обе стороны - из одной организации.

Стандарт ISO/IEC 12207 определяет организацию ЖЦ программного продукт как совокупность процессов, каждый из которых разбит на действия, состоящие из отдельных задач; устанавливает структуру(архитектуру) ЖЦ программного продукта в виде перечня процессов, действий и задач

Процессы ЖЦ ISO/IEC 12207

делятся на три группы:

1. Основные процессы.

2. Вспомогательные процессы.

предназначены для поддержки выполнения основных процессов, обеспечения качества проекта, организации верификации, проверки и тестирования ПО.

3. Организационные процессы.

определяют действия и задачи, выполняемые как заказчиком, так и разработчиком проекта для управления своими процессами.

Основные процессы

- **Приобретение(заказ)** определяет работы заказчика (организации, приобретающей программный продукт (ПП))
- **Поставка** определяет работы поставщика (организации, поставляющей ПП)
- **Разработка** определяет работы разработчика (организации, которая проектирует и разрабатывает ПП)
- **Эксплуатация** определяет работы оператора (организации, обеспечивающей обслуживание вычислительной системы в заданных условиях в интересах пользователей) - работы по внедрению ПП в эксплуатацию (конфигурирование БД и рабочих мест), обеспечение эксплуат. документацией, обучения персонала и т.д., и непосредственно эксплуатацию (локализацию проблем и устранение причин их возникновения);
- **Сопровождение** определяет работы группы сопровождения (организации, которая предоставляет услуги по сопровождению ПП – контролируемое изменение работы);

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

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

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

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

Процесс (Исполнитель процесса)	Действия	Вход	Результат
Разработка (разработчик ИС)	· Подготовка · Анализ требований к ИС · Проектирование архитектуры ИС · Разработка требований к ПО · Проектирование архитектуры ПО · Детальное проектирование ПО · Кодирование и тестирование ПО · Интеграция ПО и квалификационное тестирование ПО · Интеграция ИС и квалификационное тестирование ИС	· Техническое задание на ИС · Техническое задание на ИС, модель ЖЦ · Подсистемы ИС · Спецификации требования к компонентам ПО · Архитектура ПО · Материалы детального проектирования ПО · План интеграции ПО, тесты · Архитектура ИС, ПО, документация на ИС, тесты	· Используемая модель ЖЦ, стандарты разработки · План работ · Состав подсистем, компоненты оборудования · Спецификации требования к компонентам ПО · Состав компонентов ПО, интерфейсы с БД, план интеграции ПО · Проект БД, спецификации интерфейсов между компонентами ПО, требования к тестам · Тексты модулей ПО, акты автономного тестирования · Оценка соответствия комплекса ПО требованиям ТЗ · Оценка соответствия ПО, БД, технического комплекса и комплекта документации требованиям ТЗ

Вспомогательные процессы

- документирование;
- управление конфигурацией;
- обеспечение качества (определяет работы по обеспечению того чтобы ПП соответствовали требованиям установленным для них. Методы обеспечения качества: аттестация, верификация, аудит, совместный анализ – см. след. слайд)
- разрешение проблем (определяет процесс анализа и устранения проблем включая несоответствия независимо от их характера и источника, которые были обнаружены во время разработки, эксплуатации и др. процессов)

Методы обеспечения качества:

- **Верификация** –это процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа. Проверка позволяет оценить соответствие параметров разработки исходным требованиям. Проверка частично совпадает с тестированием, которое связано с идентификацией различий между действительными и ожидаемыми результатами и оценкой соответствия характеристик ПО исходным требованиям
- **Совместный анализ.** Работы направленные на по оценку состояния и результатов какой-либо работы. Данный процесс используется двумя любыми сторонами, когда одна из сторон (проверяющая) проверяет другую сторону (проверяемую).
- Аудит
- Аттестация

Организационные процессы

- создание инфраструктуры;
- управление;
- обучение;
- усовершенствование

Примерные вопросы

Стандарт ISO/IEC 12207. Организационные процессы

Какие организационные процессы рассматривает стандарт ISO/IEC 12207.

Содержание этих процессов

Особенности, достоинства и недостатки стандарта ISO/IEC 12207.

Стандарт ISO/IEC TR 15504

Когда был принят

Как связан со стандартом ISO/IEC 12207.

Особенности, достоинства и недостатки стандарта ISO/IEC TR 15504.

Стандарт ISO/IEC12207 разрабатывался 9 лет и достаточно быстро устарел. В 1998 г. выходит новый стандарт ISO/IECTR 15504 : Information Technology-Software Process Assessment (Оценка процессов разработки ПО). В этом документе рассматриваются вопросы аттестации, определения зрелости и усовершенствования процессов жизненного цикла ПО. Один из разделов документа содержит новую классификацию процессов жизненного цикла, являющуюся развитием стандарта ISO/IEC12207.

Стадии и этапы работы ГОСТ 34.601 -90

Соответствует каскадной модели

http://docs.nevacert.ru/files/gost/gost_34.601-1990.pdf

Российские стандарты

Существует ряд связанных ГОСТов 24 и 34 серий.

Основным является:

ГОСТ 34.601-90. Автоматизированные Системы. Стадии Создания. (введен в действие 01.01.1992 г.)

Когда речь идет о создании ИС(ПС) как правило ссылка делается на данный стандарт. Все остальные ГОСТы вызываются из архитектуры ГОСТ 34.601-90.

Стандарт устанавливает стадии и этапы создания АС (автоматизированных систем).

В **приложении 1** приведено содержание работ на каждом этапе.

В **приложении 2** перечень организаций, участвующих в работах по созданию АС

Ориентирован на каскадную модель жизненного цикла.

Российские стандарты

ГОСТ 34.602–89 устанавливает **состав, содержание, правила оформления, порядок разработки, согласования и утверждения документа “Техническое задание на создание (развитие или модернизацию) автоматизированной системы”.**

Проект ТЗ разрабатывает организация-разработчик системы с участием заказчика на основании технических требований (заявки, тактико-технического задания и т.п.). При необходимости, разработчик ТЗ и заказчик осуществляют согласование проекта ТЗ с органами государственного надзора и заинтересованными организациями.

Утверждение ТЗ осуществляют руководители предприятий (организаций) разработчика и заказчика ИС.

Российские стандарты

- **ГОСТ 34.201–89** определяет виды, наименования, комплектность и обозначение документов, разрабатываемых на стадиях создания ИС (в общем случае),
- **РД 50-34.698-90** – содержит требования к содержанию этих документов.
- **РД 50-680-88** – методические указания и устанавливают назначение, состав, основные принципы создания и функционирования АС.

Общесистемные принципы создания ИС

РД 50-680-88

Общесистемные принципы создания ИС определены РД 50-680-88

Создание ИС должно осуществляться на основе принципов:

- системности,
- совместимости,
- стандартизации (унификации),
- развития (открытости)
- эффективности.

Принцип системности

заключается в том, что на всех стадиях создания и развития целостность системы должна обеспечиваться связями между подсистемами и комплексами задач. Требования к создаваемой системе должны определяться со стороны высшей по иерархии системы, включающей в себя данную ИС. В создаваемой ИС должна быть обеспечена связность решения отдельных автоматизированных задач системы и работы учреждения в целом. Этот принцип должен быть реализован путем создания единой подсистемы управления ИС.

Принцип совместимости

заключается в том, что при создании ИС должны быть реализованы информационные интерфейсы, благодаря которым она может взаимодействовать с другими ИС в соответствии с установленными правилами.

Символы, коды, информационные и технические характеристики структурных связей между подсистемами и компонентами ИС должны быть согласованы так, чтобы обеспечивалось совместное функционирование всех подсистем ИС.

В ИС должны использоваться единые термины, символы, усл. обозначения и способы представления информации во всех автоматизированных задачах, комплексах задач, подсистемах.

Этот принцип требует использования в ИС единой системы классификации и кодирования информации, единых правил сопоставления всех взаимосвязанных информационных показателей.

Принцип стандартизации (унификации)

состоит в том, что подсистемы и компоненты системы должны быть по возможности типовыми. Этот принцип должен реализовываться путем:

- создания единой базы данных;
- использования единого информационного обеспечения;
- унификации алгоритмов решения задач, программных модулей, программ и т.п.

Принцип развития (открытости)

состоит в том, что ИС должна создаваться как развивающаяся система, допускающая пополнение, совершенствование и обновление подсистем и компонентов. Этот принцип должен реализовываться за счет открытой структуры всех подсистем ИС. Развитие системы будет осуществляться путем пополнения ее новыми подсистемами и компонентами, модернизации действующих подсистем и компонентов, обновления используемых средств вычислительной техники более совершенными.

Принцип эффективности

заключается в достижении рационального соотношения между затратами на создание ИС и целевыми эффектами, включая конечные результаты, получаемые в результате автоматизации.

Этапы и фазы ГОСТ 34.601-90:

Этап	Фаза
1 Формирование требований	1.1 Обследование объекта 1.2 Формирование требований пользователя к системе 1.3.Оформление отчета о выполненной работе и заявки на разработку АС (тактико-техническое задание)
2 Разработка концепции АС	2.1. Изучение объекта. 2.2. Проведение необходимых научно-исследовательских работ. 2.3. Разработка вариантов концепции АС, удовлетворяющего требованиям пользователя. 2.4. Оформление отчёта о выполненной работе.
3 Техническое задание	3.1 Разработка и утверждение **технического задания** на создание системы

http://www.prj-exp.ru/gost/gost_34-601-90.php

Этапы и фазы ГОСТ 34.601-90:

Этап	Фаза
4 Эскизный проект	4.1. Разработка предварительных проектных решений по системе и её частям. 4.2. Разработка документации на АС и её части.
5 Технический проект	5.1 Разработка проектных решений по системе и ее частям 5.2 Разработка документации на систему и ее части 5.3 Разработка и оформление документации на поставку изделий для комплектования ИС и/или технических требований (технических заданий) на их разработку 5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации.

Этапы и фазы ГОСТ 34.601-90:

Этап	Фаза
6 Рабочая документация	6.1 Разработка рабочей документации на систему и ее части 6.2. Разработка или адаптация программ
7 Ввод в действие	7.1 Подготовка объекта автоматизации к вводу системы в действие 7.2 Подготовка и обучение персонала 7.3 Комплектация поставляемыми изделиями 7.4 Строительно-монтажные работы 7.5 Пуско-наладочные работы 7.6 Проведение опытных испытаний 7.7 Проведение опытной эксплуатации 7.8 Проведение приемочных испытаний
8 Сопровождение системы	8.1 Выполнение работ в соответствии с гарантийными обязательствами 8.2 Послегарантийное обслуживание

Техническое задание

- это документ, определяющий цели, требования и основные исходные данные, необходимые для разработки автоматизированной системы управления

Задачи, решаемые при составлении ТЗ:

- установить общую цель создания ИС, определить состав подсистем и функциональных задач
- разработать и обосновать требования, предъявляемые к подсистемам
- определить перечень задач создания системы и исполнителей
- определить этапы создания системы и сроки их выполнения

Типовые требования к содержанию ТЗ (разделы ТЗ) ГОСТ 34.602- 89

(http://www.rugost.com/index.php?catid=22&id=96:gost-34602-89&Itemid=53&option=com_content&view=article)

- Общие сведения
- Назначение и цели создания (развития) системы
- Характеристика объектов автоматизации
- Требования к системе
- Состав и содержание работ по созданию системы
- Порядок контроля и приемки системы
- Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие
- Требования к документированию
- Источники разработки

+ ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению http://www.prj-exp.ru/gost/gost_19-201-78.php

Технический проект

- это техническая документация, содержащая общесистемные проектные решения, алгоритмы решения задач, а также оценку экономической эффективности автоматизированной системы управления и перечень мероприятий по подготовке объекта к внедрению

По ГОСТ 34.601-90 допускается объединять стадии «Технический проект» и «Рабочая документация» в одну стдию «Технорабочий проект»

Типовые требования к содержанию ТП (разделы ТП)

- Пояснительная записка
- Функциональная и организационная структура системы
- Постановка задач и алгоритмы решения
- Организация информационной базы
- Альбом форм документов
- Система математического обеспечения
- Принцип построения комплекса технических средств
- Расчет экономической эффективности системы
- Мероприятия по подготовке объекта к внедрению системы
- Ведомость документов

Состав и содержание технического проекта определяется *РД 50-34.698-90 и ГОСТ 2.106*

Кратко про некоторые другие стандарты

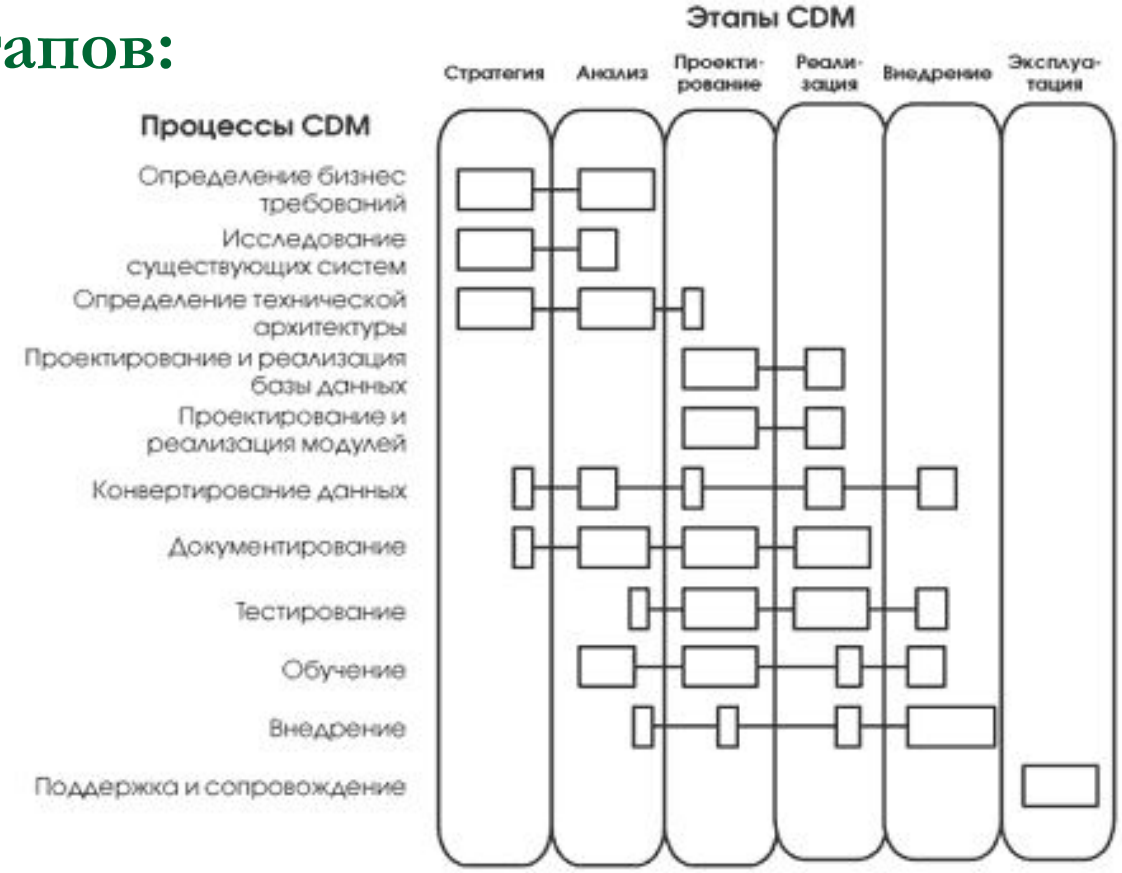
CDM корпорации Oracle (www.oracle.com)

CDM – это составная часть глобальной методологии разработки ИС — Oracle Method - комплекс методов, охватывающий большинство процессов ЖЦ ПО.

CDM может быть применен при разработке ИС разных типов и с использованием разных подходов, как самостоятельно так и в качестве составной части другого метода.

CDM - это технологический материал, детализированный до уровня заготовок проектных документов, рассчитанных на использование в проектах с применением Oracle.

В соответствии с CDM ЖЦ ПО формируется из определенных этапов (фаз) проекта и процессов, каждый из которых выполняется в течение нескольких этапов:



Подробнее см. статья **Современные технологии создания программного обеспечения. Обзор.** А.М. Вендров , http://citforum.ru/programming/application/program/3.shtml

Варианты подходов в CDM

CDM предусматривает возможность выбора подхода к разработке.

При выборе подхода к разработке:

- оценивается масштаб, степень сложности и критичность будущей системы.
- учитываются стабильность требований, сложность и количество бизнес-правил, количество автоматически выполняемых функций, разнообразие и количество пользователей, степень взаимодействия с другими системами, критичность приложения для основного бизнес-процесса компании и целый ряд других.

Подходы к разработке ПО(по CDM)

Классический подход (classic)

Рис. См. выше

- для наиболее сложных и масштабных проектов;
- предусматривает последовательный и детерминированный порядок выполнения задач.

Сроки – от 8 до 36 месяцев

Подход быстрой разработки (Fast Track)

Предусматривает 4 этапа:

1 - стратегия,

2 - моделирование требований,

3 – проект-ие и генерация ИС

4 - внедрение в эксплуатацию. Используется для реализации небольших и средних проектов с несложной архитектурой системы, гибкими сроками и четкой постановкой задач.

Сроки от 4 до 16 месяцев.

Rational Unified Process (RUP)

Одна из наиболее совершенных технологий, претендующих на роль мирового корпоративного стандарта – создана компанией Rational Software.

Принципы

1 - Итерационный и инкрементный (наращиваемый) подход к созданию ПО. **(определяющий)**

2 - Планирование и управление проектом на основе функциональных требований к системе - вариантов использования.

3 - Построение системы на базе архитектуры ПО.

В соответствии с 1 принципом разработка системы выполняется в виде нескольких краткосрочных мини-проектов фиксированной длительности (от 2 до 6 недель), называемых итерациями.

RUP – итеративная модель разработки

Каждая итерация(цикл) включает свои собственные этапы

1 - анализ требований,

2 - проектирования,

3 - реализации,

4 - тестирования,

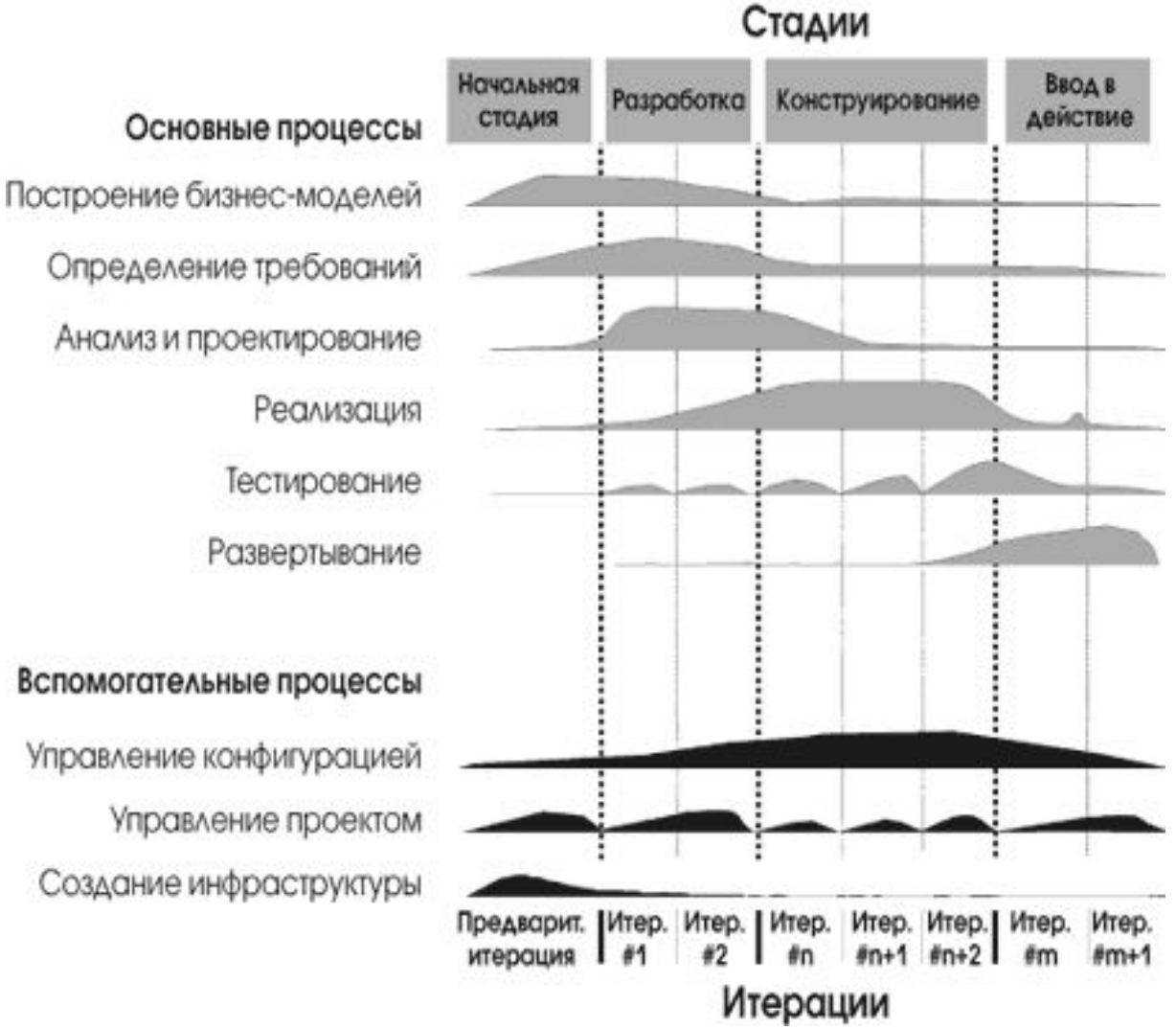
5 - интеграции

и завершается созданием работающей системы.

Каждая итерация (цикл) завершается генерацией версии системы. Если после этого работа над проектом не прекращается, то полученный продукт продолжает развиваться и идут те же фазы.

Суть работы в рамках RUP это создание и **сопровождение моделей, а не бумажных документов, поэтому этот процесс привязан к использованию конкретных средств моделирования (унифицированного языка моделирования UML), также конкретной технологии проектирования и разработки.**

Графическое представление RUP



Подробнее см. статья **Современные технологии создания программного обеспечения. Обзор.** А.М. Вендров , http://citforum.ru/programming/application/program/3.shtml

Особенности RUP

- Ранняя идентификация и непрерывное (до окончания проекта) устранение основных рисков.
- Концентрация на выполнении требований заказчиков к исполняемой программе (анализ и построение модели прецедентов (вариантов использования)).
- Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки.
- Компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта.
- Постоянное обеспечение качества на всех этапах разработки проекта (продукта).
- Работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам.

RUP и другие методологии разработки ПО. *Алексей Закис*
http://www.shmakov.ru/news/text136.html

MSF – методология разработки ПО, предложенная Microsoft(с 1994 г.)

- опирается на практический опыт Microsoft и описывает управление людьми и рабочими процессами в процессе разработки решения.
- - похожа на RUP, также есть 4 стадии + итеративный подход

MSF фокусируется на следующих аспектах:

- согласование деловых и технологических целей;
- определение четких целей, ролей и ответственностей для проекта;
- реализация итеративного процесса на основе вех и контрольных точек;
- упреждающее управление рисками;
- эффективная реакция на изменения.

Подробнее см. Статью: MSF – философия создания IT-решений или голые амбиции лидера *Сергей Якимчук, Softline Company (Киев)*
http://citforum.ru/SE/project/msf/

XP - экстремальное программирование

“XP - это ответ сообщества программистов на наступление формальных подходов к созданию программных продуктов и призвано вернуть в среду разработчиков дух творчества.”
“Неоправданное, гипертрофированное внимание к Процессу в ущерб другим факторам разработки породило массу формальных процедур — но качество полученных продуктов и количество успешных проектов оказалось разочаровывающим.”

Источник: статья “Экстремальное программирование и быстрая разработка ПО” Арсений Чеботарев
http://www.realcoding.net/articles/ekstremalnoe-programmirovanie-i-bystraya-razrabotka-po.html

XP – гибкая методология

- разработана Kent Beck, Ward Cunningham, и Ron Jeffries,
- на сегодня является наиболее известной из гибких методологий
- XP проповедует коммуникабельность, простоту, обратную связь и отвагу.
- Интерес к XP рос снизу вверх, от разработчиков и тестировщиков, замученных тягостным процессом, документацией и пр.
- Практически все гибкие методологии используют итеративный подход, при котором детально планируется только ограниченный объем работ, связанный с выпуском очередного релиза.
- Практически все гибкие методологии ориентированы на максимально неформальный подход к разработке. Если проблему можно решить в разговоре, то лучше именно так и сделать. Причем оформлять принятое решение в виде бумажного или электронного документа нужно только тогда, когда без этого невозможно обойтись.

XP описывается как следующие практики:

- игра в планирование,
- короткие релизы,
- метафоры,
- простой дизайн,
- переработки кода (refactoring),
- разработка "тестами вперед",
- парное программирование,
- коллективное владение кодом,
- 40-часовая рабочая неделя,
- постоянное присутствие заказчика и
- стандарты кода.

Основа экстремального программирования

- **Экстремально короткий цикл (короткие релизы)**

очень короткий, постоянно повторяющийся цикл разработки, составляющий одну-три недели.

К концу каждого цикла имеется полностью рабочий, функциональный и протестированный релиз приложения. Эти циклы должны быть повторяющимися и бесперебойными на протяжении всего проекта.

- **Рефракторинг**

Это наведение порядка в коде, переработка отдельных файлов и их групп с целью наведения порядка, удаления максимального количества ненужных фрагментов, объединения классов на основе схожей функциональности, коррекция комментариев, осмысленное переименование объектов и так далее.

- **Кодирование в глубину**

означает, что в течение цикла должна быть полностью разработана и протестирована отдельная функциональность и проигнорированы соседние с ней области.

Выводы по использованию XP

- тщательное предварительное проектирование ПО заменяется, с одной стороны, постоянным присутствием в команде заказчика, готового ответить на любой вопрос и оценить любой прототип, а с другой стороны, регулярными переработками кода (так называемый рефакторинг).
- основой проектной документации считается тщательно прокомментированный код.
- очень большое внимание уделяется тестированию.

Источник: статья “Экстремальное программирование и быстрая разработка ПО” Арсений Чеботарев
http://www.realcoding.net/articles/ekstremalnoe-programmirovanie-i-bystraya-razrabotka-po.html

Требуемый уровень формализма?

А сколько формализма нужно?

- Долгое время бытовало мнение, что **чем больше тщательно оформленной документации выпускается в ходе выполнения проекта - тем лучше.**
- Значит, тем тщательнее проработан проект, тем полнее сохраняются и передаются в группу сопровождения знания и проектные решения по проекту и тем проще в дальнейшем сопровождать продукт.
- Однако оказалось, что это не совсем так.

Основные факторы влияющие на оптимальный уровень формализма:

- **Масштаб проекта.** Чем больше людей участвует в проекте, тем более формально он, как правило, должен вестись.
- **Критичность проекта.** Проекты, в которых ошибки могут приводить к тяжелым последствиям вплоть до гибели людей, должны проводиться существенно более формально, чем те проекты, в которых ошибки приводят только к временным неудобствам.
- **Распределение участников.** Чем компактнее расположены участники, чем легче им общаться между собой, тем менее формализованным может быть проект.
- **Новизна проекта.** Чем более новые для разработчиков технологии вы используете, чем меньше вы знакомы с предметной областью, тем боле тщательно надо прорабатывать проектные решения, и, соответственно, тем более формально это должно происходить.

Основные факторы влияющие на оптимальный уровень формализма:

- **Требования заказчика.** Они существенно различаются в зависимости от отрасли и статуса организации-заказчика. Как правило, эти требования выше для госпредприятий.
- **Ожидаемая долговечность проекта.** Если разрабатываемое для конкретного заказчика ПО предполагается через пару лет будет заменено новым, то вряд ли есть смысл тратить много сил на документацию, которая могла бы значительно удешевить более длительный процесс сопровождения ПО. Зато если срок жизни проекта предполагается достаточно длинным - без хорошей документации не обойтись.

Примерные вопросы к темам 5-7

- Причины создания – основные идеи
- Особенности
- Достоинства
- Недостатки
- Авторы(компании)
- Применяемые методы, подходы
- Стадии, этапы
- Примеры использования – отзывы