Программная инженерия

Лекция 2

Лекция 2 Процесс разработки ПО

Цели:

1. Знать основные концепции , лежащие **в основе процесса создания ПО и моделей этого процесса**.
2. Знать какие существуют модели процесса создания ПО и когда
   каждая из них используется.
3. Знать схему построения моделей формирования требований:
   к ПО, к разработке ПО, к тестированию ПО, к модернизации ПО.
4. Иметь понятие о CASE-технологиях поддержки создания ПО.

Читать по Лекции 2

Книга	**Страницы**
Д.В.*Кознов ,и др*. *Введение в программную инженерию*	***Лекция 2:*** ***Стр. 7-13***
Иан Соммервилл Инженерия программного обеспечения	***Введение :*** ***Стр. 53-79***

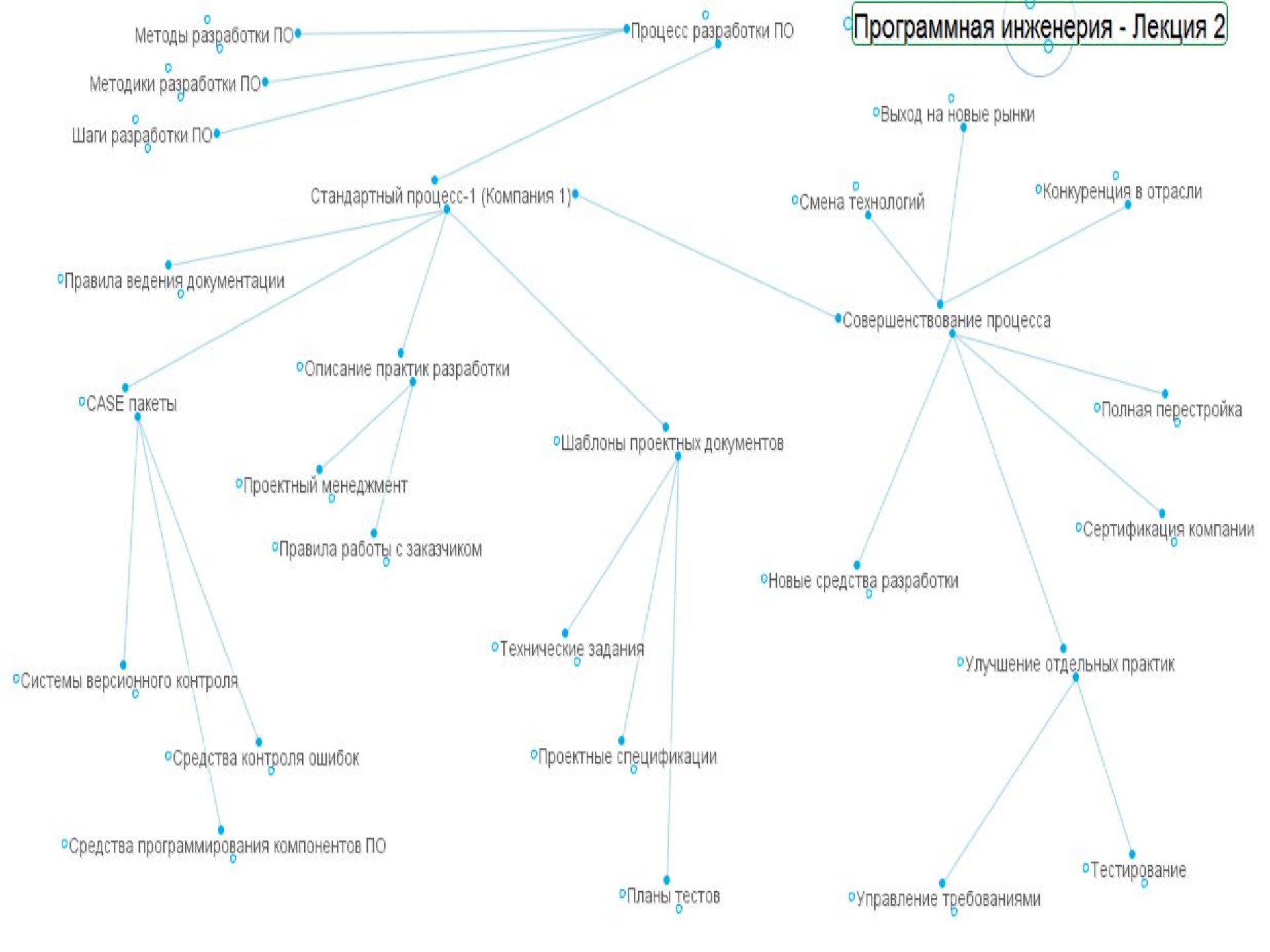
Процесс разработки ПО

Центральным объектом изучения программной инженерии является ***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.).

На сегодняшний день не существует ***универсального процесса*** разработки ПО

Каждый текущий процесс разработки, осуществляемый некоторой командой в рамках определенного проекта, имеет большое количество особенностей и индивидуальностей.

Однако в рамках компании возможна и полезна объединение и стандартизация всех текущих процессов, которую будем называть **стандартным процессом**


Программная инженерия - Лекция 2
Методы разработки ПО
Процесс разработки ПО
Методики разработки ПО
Шаги разработки ПО
Стандартный процесс-1 (Компания 1)
Правила ведения документации
Описание практик разработки
CASE пакеты
Шаблоны проектных документов
Проектный менеджмент
Правила работы с заказчиком
Системы версионного контроля
Средства контроля ошибок
Средства программирования компонентов ПО
Технические задания
Проектные спецификации
Планы тестов
Выход на новые рынки
Смена технологий
Конкуренция в отрасли
Совершенствование процесса
Полная перестройка
Сертификация компании
Новые средства разработки
Улучшение отдельных практик
Тестирование
Управление требованиями

Совершенствование процесса разработки ПО

Главная трудность реального совершенствования процессов в компании заключается в том, что она при этом должна работать и создавать ПО, ее нельзя «закрыть на учет».

Отсюда вытекает идея непрерывного улучшения процесса, так сказать, малыми порциями, чтобы не так болезненно. Это тем более разумно, что новые технологии разработки, появляющиеся на рынке, а также развитие уже существующих нужно постоянно отслеживать.

Pull/Push стратегии

Pull стратегия – инновации нацелены на решение конкретных проблем компании

Technology push – широкомасштабное внедрение инноваций из стратегических соображений.

Классические модели процесса разработки ПО

Модель процесса – формализованное описание динамики развития тех или иных видов деятельности, образующих процесс

Фазы и виды деятельности. Говоря о моделях процессов, необходимо различать фазы и виды деятельности.

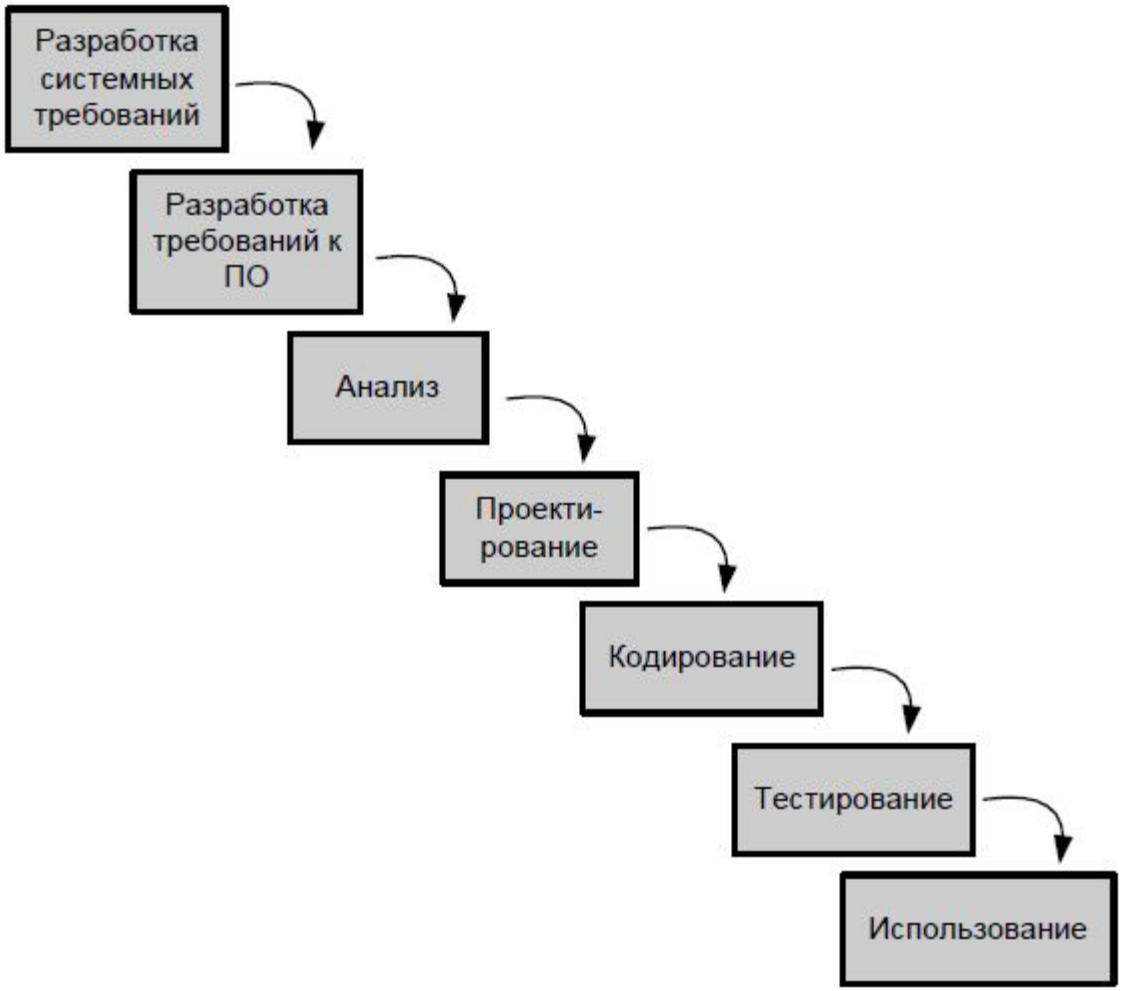
Фаза (phase) – это определенный этап процесса, имеющий начало, конец и выходной результат. Например, фаза проверки осуществимости проекта, сдачи проекта и т.д. Фазы следуют друг за другом в линейном порядке, характеризуются предоставлением отчетности заказчику и, часто, выплатой денег за выполненную часть работы.

Вид деятельности (activity) – это определенный тип работы, выполняемый в процессе разработки ПО

В рамках одной фазы может выполнятся много различных видов деятельности

Водопадная (каскадная) модель

была предложена в 1970 году Винстоном Ройсом. Фактически, впервые в процессе разработки ПО были выделены различные шаги разработки.



Недостатки водопадной модели:

• отождествление фаз и видов деятельности, что влечет потерю гибкости разработки, в частности, трудности поддержки итеративного процесса разработки;

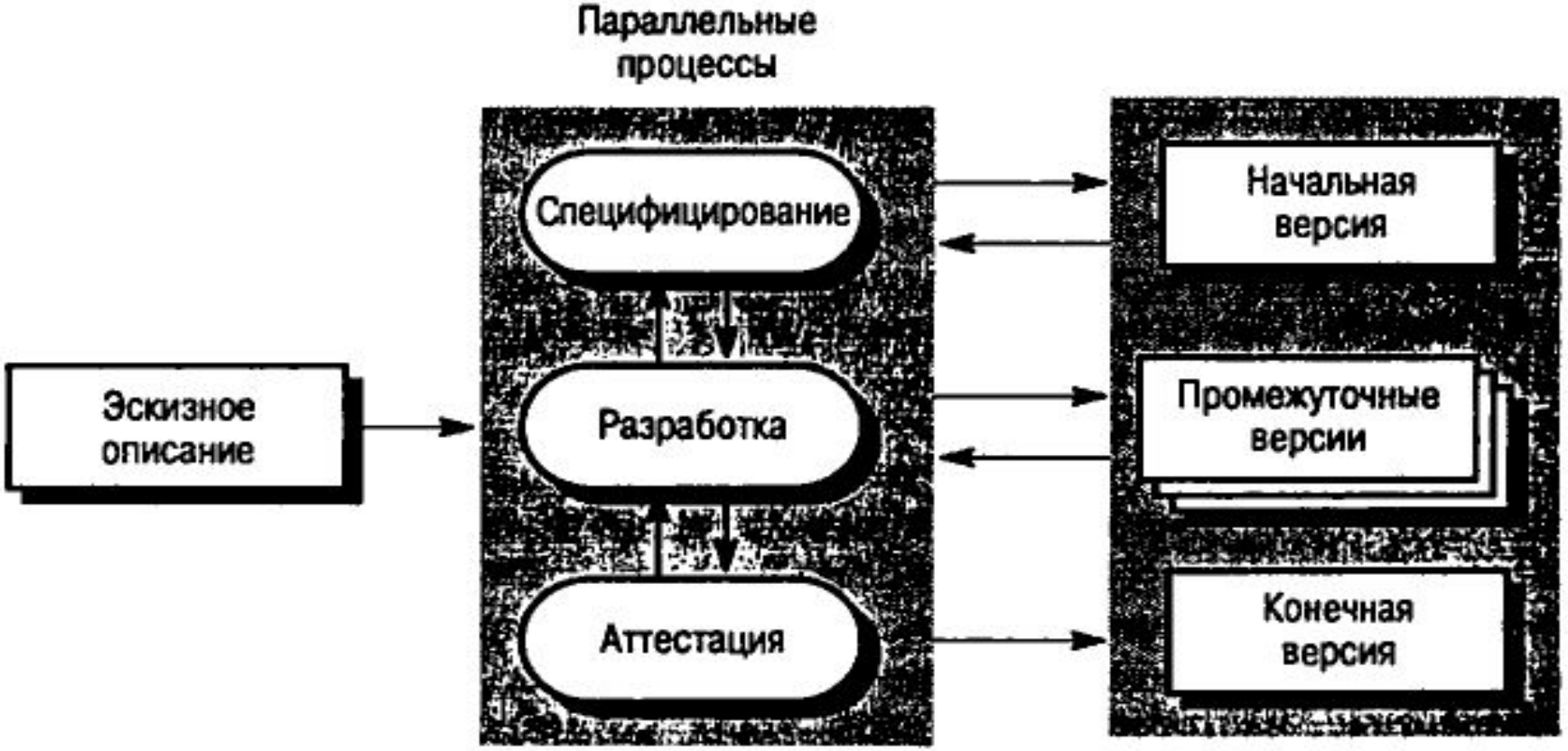
• требование полного окончания фазы-деятельности, закрепление результатов в виде подробного исходного документа (технического задания, проектной спецификации), одного опыта разработки ПО, показывает, что невозможно полностью зафиксировать требования, дизайн системы и т.д. – все это подвержено изменениям; и причины тут не только в том, что подвижно окружение проекта, но и в том, что заранее не удается точно определить и сформулировать многие решения, они проясняются и уточняются лишь впоследствии;

• интеграция всех результатов разработки происходит в конце, вследствие чего интеграционные проблемы дают о себе знать слишком поздно;

• пользователи и заказчик не могут ознакомиться с вариантами системой во время разработки, и видят результат только в самом конце; тем самым, они не могут повлиять на процесс создания системы, и поэтому увеличиваются риски непонимания между разработчиками и пользователями/заказчиком;

• модель неустойчива к сбоям в финансировании проекта или перераспределению денежных средств, начатая разработка, фактически, не имеет альтернатив “по ходу дела”

Эволюционная модель

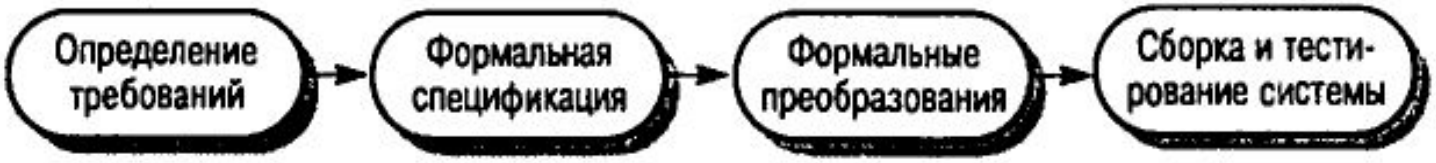


Эволюционная модель разработки

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 (рис. 3.2). Отличительной чертой данной модели является то, что процессы спецификирования, разработки и аттестации ПО выполняются параллельно при постоянном обмене информацией между ними.

Формальная разработка систем

Этот подход к созданию ПО имеет много черт, сходных с каскадной моделью, но построен на основе формальных математических преобразований системной спецификации в исполняемую программу. Процесс создания программного обеспечения в соответствии с этим подходом показан на рис. 3.3. Здесь для упрощения не указаны обратные связи между этапами процесса.



Модель формальной разработки ПО

Между данным подходом и каскадной моделью существуют следующие кардинальные отличия.

1. Здесь спецификация системных требований имеет вид детализированной формальной спецификации, записанной с помощью специальной математической нотации.
2. Процессы проектирования, написания программного кода и тестирования системных модулей заменяются процессом, в котором формальная спецификация путем последовательных формальных преобразований трансформируется в исполняемую программу. Этот процесс показан на рис.

Разработка ПО на основе ранее созданных компонентов

Этот подход основан на наличии большой базы существующих программных компонентов, которые можно интегрировать в создаваемую новую систему. Часто такими компонентами являются свободно продаваемые на рынке программные продукты, которые можно использовать для выполнения определенных специальных функций, таких как форматирование текста, числовые вычисления и т.п. Общая модель процесса разработки ПО с повторным использованием ранее созданных компонентов показана на рис. 3.5.

Неформальное решение о повторном использовании ранее созданных программных компонентов обычно принимается независимо от общего процесса создания ПО. Вместе с тем на протяжении нескольких последних лет все более широко применяется подход к созданию ПО, основанный именно на повторном использовании ранее созданных программных модулей.

Рис. 3.5. Разработка ПО с повторным использованием ранее созданных компонентов

Модель пошаговой

Модель пошаговой разработки была предложена Миллсом (Mills, [240]) как попытка уменьшить количество повторно выполняемых работ в процессе создания ПО и увеличить для заказчика временной период окончательного принятия решения обо всех деталях системных требований.

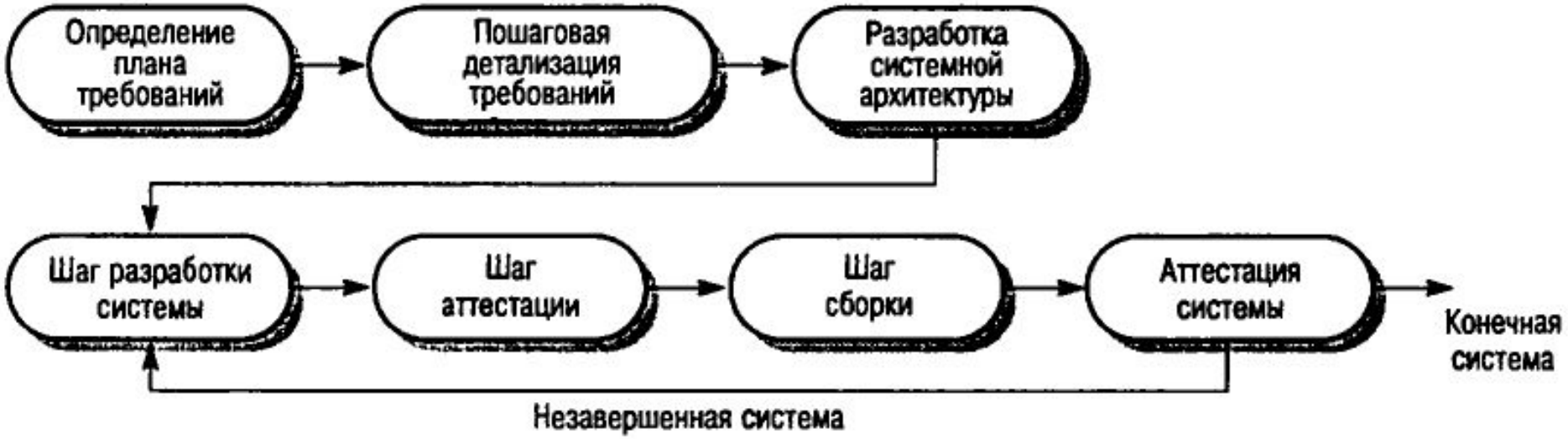


Рис. 3.6. Модель пошаговой разработки

В процессе пошаговой разработки заказчик сначала в общих чертах определяет те сервисы (функциональные возможности), которые должны присутствовать у создаваемой системы. При этом устанавливаются приоритеты, т.е. определяется, какие сервисы более важны, а какие — менее. Также определяется количество шагов разработки, причем на каждом шаге должен быть получен системный компонент, реализующий определенное подмножество системных функций. Распределение реализации системных сервисов по шагам разработки зависит от их приоритетов. Сервисы с более высокими приоритетами реализуются первыми.

Процесс пошаговой разработки имеет целый ряд достоинств.

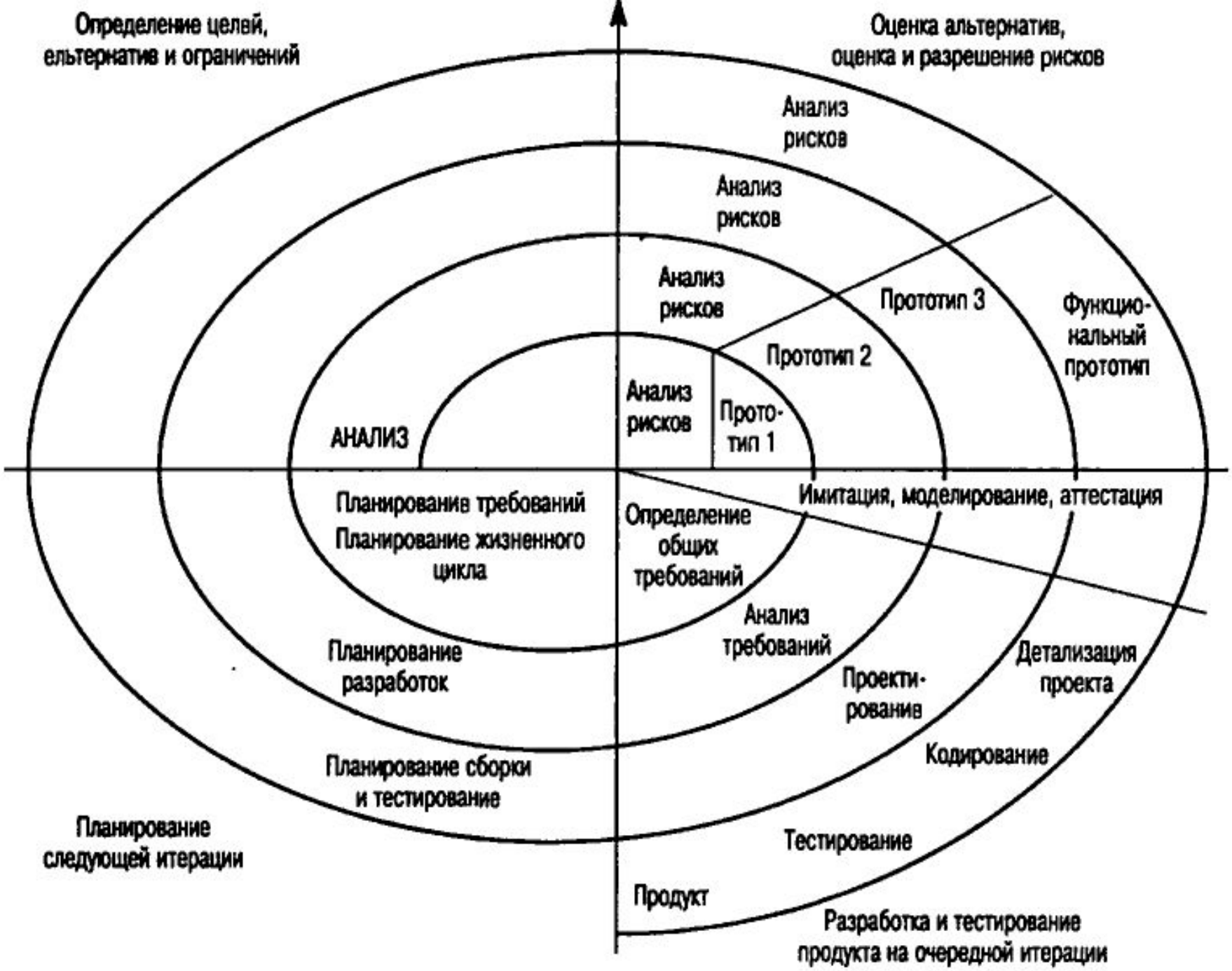
1. Заказчику нет необходимости ждать полного завершения разработки системы, чтобы получить о ней представление. Компоненты, полученные на первых шагах разработки, удовлетворяют наиболее критическим требованиям (так как имеют наибольший приоритет) и их можно оценить на самой ранней стадии создания системы.
2. Заказчик может использовать компоненты, полученные на первых шагах разработки, как прототипы и провести с ними эксперименты для уточнения требований к тем компонентам, которые будут разрабатываться позднее.
3. Данный подход уменьшает риск общесистемных ошибок. Хотя в разработке отдельных компонентов возможны ошибки, но эти компоненты должны пройти соответствующее тестирование и аттестацию, прежде чем их передадут заказчику.
4. Поскольку системные сервисы с высоким приоритетом разрабатываются первыми, а все последующие компоненты интегрируются с ними, неизбежно получается так, что наиболее важные подсистемы подвергаются более тщательному всестороннему тестированию и проверке. Это значительно снижает вероятность программных ошибок в особо важных частях системы.

Спиральная модель процесса разработки ПО

Спиральная модель была предложена Бэри Боемом в 1988 году для преодоления недостатков водопадной модели. Разработка продукта осуществляется по спирали, каждый виток которой является определенной фазой разработки, в рамках которой может осуществляться много различных видов деятельности.

Каждый виток имеет следующую структуру (секторы):

- определение целей, ограничений и альтернатив проекта;
- оценка альтернатив, оценка и разрешение рисков; возможно использование прототипирования , симуляция системы, визуальное моделирование и анализ спецификаций; фокусировка на самых рисковых частях проекта;
- разработка и тестирование – здесь возможна водопадная или иные модели и методы разработки ПО;
- 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 ПО


Определение целей,
ельтернатив и ограничений
Оценка альтернатив,
оценка и разрешение рисков
Анализ
рисков
Анализ
рисков
Анализ
рисков
Прототип 3
Функцио-
нальный
прототип
Прототип 2
Анализ
рисков
Прото-
тип 1
АНАЛИЗ
Планирование требований
Планирование жизненного
цикла
Имитация, моделирование, аттестация
Определение
общих
требований
Анализ
требований
Планирование
разработок
Проекти-
рование
Детализация
проекта
Кодирование
Планирование сборки
и тестирование
Планирование
следующей итерации
Тестирование
Продукт
Разработка и тестирование
продукта на очередной итерации

Спецификация Программного обеспечения

Схема процесса разработки требований показана на рис. 3.8. Результатом его выполнения является разработка документации, формализующей требования, предъявляемые к системе, т.е. создание системной спецификации. В этой документации требования обычно представлены на двух уровнях детализации. На самом верхнем уровне представлены требования, определяемые конечными пользователями или заказчиками ПО; но для разработчиков необходима более детализированная системная спецификация.

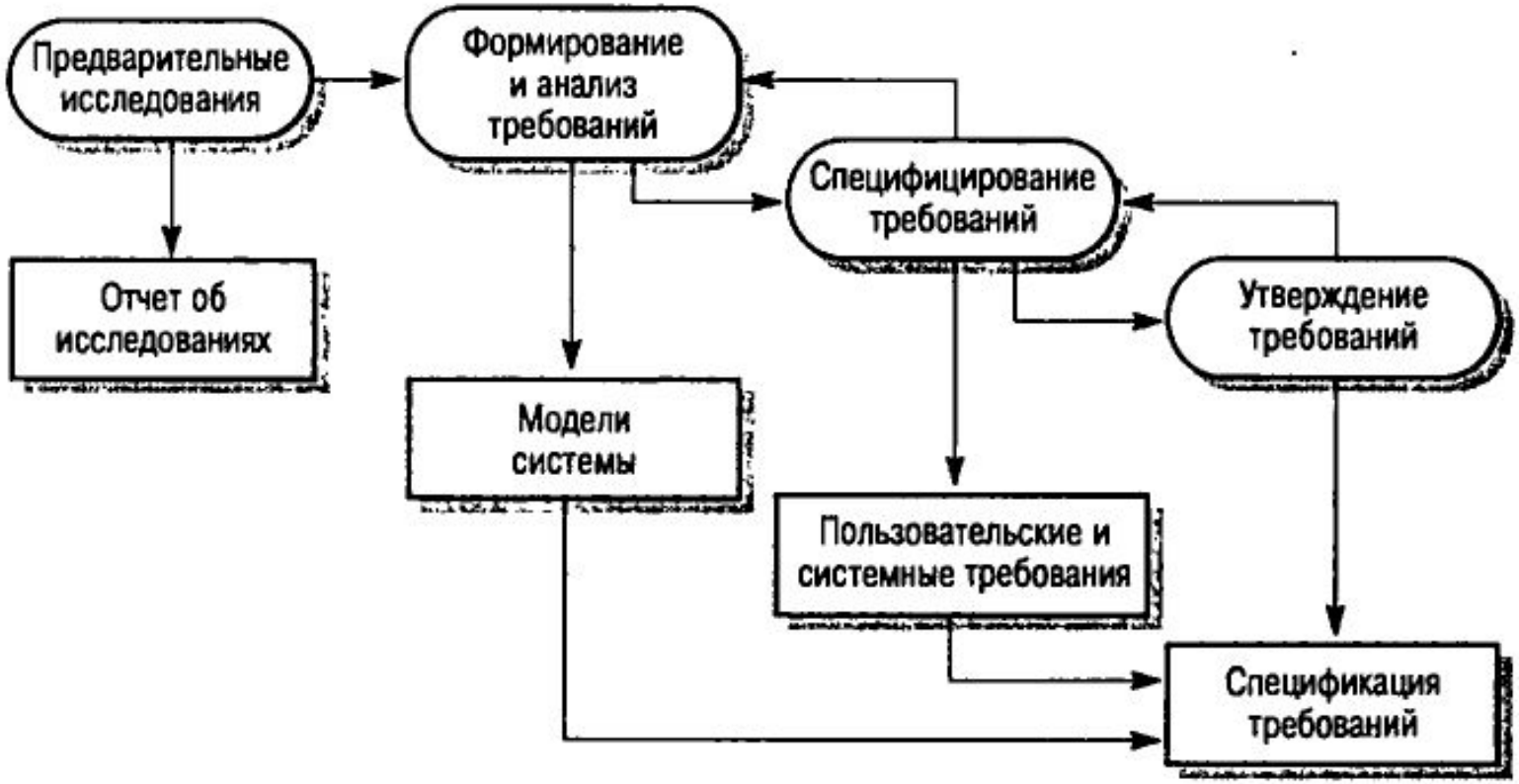


Рис. 3.8. Процесс разработки требований

Процесс разработки требований включает четыре основных этапа.

1. *Предварительные исследования.* Оценивается степень удовлетворенности пользователей существующими программными продуктами и аппаратными средствами, а также экономическая эффективность будущей системы и возможность уложиться в существующие бюджетные ограничения при ее разработке. Этот этап должен быть по возможности коротким и дешевым.
2. *Формирование и анализ требований.* Формируются системные требования путем изучения существующих аналогичных систем, обсуждения будущей системы с потенциальными пользователями и заказчиками, анализа задач, которые должна решать система, и т.п. Этот этап может включать разработку нескольких моделей системы и ее прототипов, что помогает сформировать функциональные требования к системе.
3. *Специфицирование требований.* Осуществляется перевод всей совокупности информации, собранной на предыдущем этапе, в документ, определяющий множество требований. Этот документ обычно содержит два типа требований: пользовательские — обобщенные представления заказчиков и конечных пользователей о системе; системные — детальное описание функциональных показателей системы.
4. *Утверждение требований.* Проверяется выполнимость, согласованность и полнота множества требований. В процессе формирования ограничений неизбежно возникновение каких-либо ошибок. На этом этапе они должны быть по возможности выявлены и устранены.

Проектирование и реализация ПО

Рис. 3.9. Обобщенная схема процесса проектирования

Отдельные этапы проектирования ПО

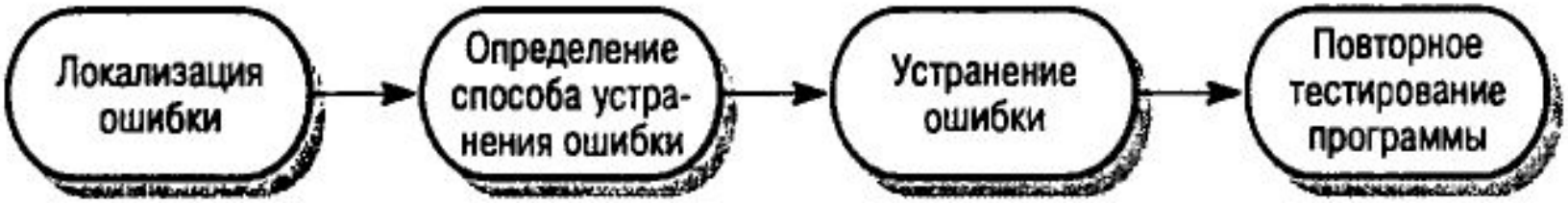
1. *Архитектурное проектирование.* Определяются и документируются подсистемы и взаимосвязи между ними.
2. *Обобщенная спецификация.* Для каждой подсистемы разрабатывается обобщенная спецификация на ее сервисы и ограничения.
3. *Проектирование интерфейсов.* Для каждой подсистемы определяется и документируется ее интерфейс. Спецификацин на эти интерфейсы должны быть точно выраженными и однозначными, чтобы использование подсистем не требовало знаний о том, как они реализуют свои функции. На этом этапе можно применить методы формальных спецификаций, рассмотренные в главе 9.
4. *Компонентное проектирование.* Проводится распределение системных функций (сервисов) по различным компонентам и их интерфейсам.
5. *Проектирование структур данных.* Детально разрабатываются структуры данных, необходимые для реализации программной системы.
6. *Проектирование алгоритмов.* Детально разрабатываются алгоритмы, предназначенные для реализации системных сервисов.

Методы проектирования ПО

Наиболее разработанным подходом к проектированию ПО обладают так называемые структурные методы, которые предлагают мно́жество формализованных нотаций и нормативных руководств для проектирования программных продуктов.

Применение структурных методов обычно приводит к созданию графических моделей системы и большому объему проектной документации. CASE-средства предназначены для поддержкн именно таких методов. Структурные методы успешно применялись во многих программных проектах. Они значительно снижают стоимость разработки, поскольку используют стандартные нотации для получення стандартной проектной документации. Ни об одном из этих методов нельзя сказать, что он лучше или хуже других. Успешное или неуспешное применение того или иного метода часто зависит от типа разрабатываемого ПО.

Программирование и отладка



Аттестация программных

1. *Тестирование компонентов.* Тестируются отдельные компоненты для проверки правильности их функционирования. Каждый компонент тестируется независимо от других.

2. *Тестирование модулей.* Программный модуль – это совокупность зависимых компонентов, таких как описание класса объектов, декларирование абстрактных типов данных и набор процедур и функций. Каждый модуль тестируется независимо от других системных модулей.

3. *Тестирование подсистем.* Тестируются наборы модулей, которые составляют отдельные подсистемы. Основная проблема, которая часто проявляется на этом этапе, – несогласованность модульных интерфейсов. Поэтому при тестировании подсистем основное внимание уделяется обнаружению ошибок в модульных интерфейсах путем прогона их через все возможные режимы.

4. *Тестирование системы.* Из подсистем собирается конечная система. На этом этапе основное внимание уделяется совместимости интерфейсов подсистем и обнаружению программных ошибок, которые проявляются в виде непредсказуемого взаимодействия между подсистемами. Здесь также проводится аттестация системы, т.е. проверяется соответствие системной спецификации ее функциональных и нефункциональных показателей, а также оцениваются интеграционные характеристики системы.

5. *Приемочные испытания.* Это конечный этап процесса тестирования, после которого система принимается к эксплуатации. Здесь система тестируется с привлечением данных, предоставляемых заказчиком системы, а не на основе тестовых данных, как было на предыдущем этапе.