Лекция 11
БГТУ, ФИТ, 1 семестр

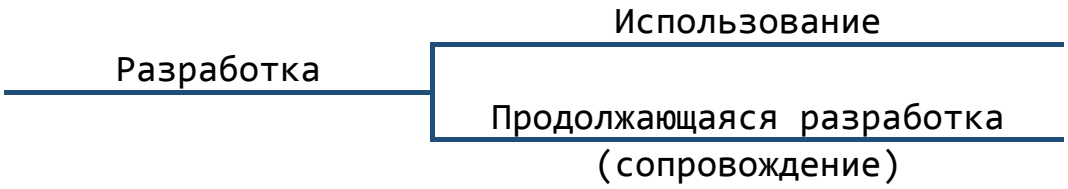
Основы программной инженерии (ПИ)

Жизненный цикл разработки программного обеспечения

План лекции:
- понятие жизненного цикла разработки ПО;
- основные определения;
- взаимосвязь между процессами жизненного цикла;
- стандарт ISO/IEC 12207: 1995.

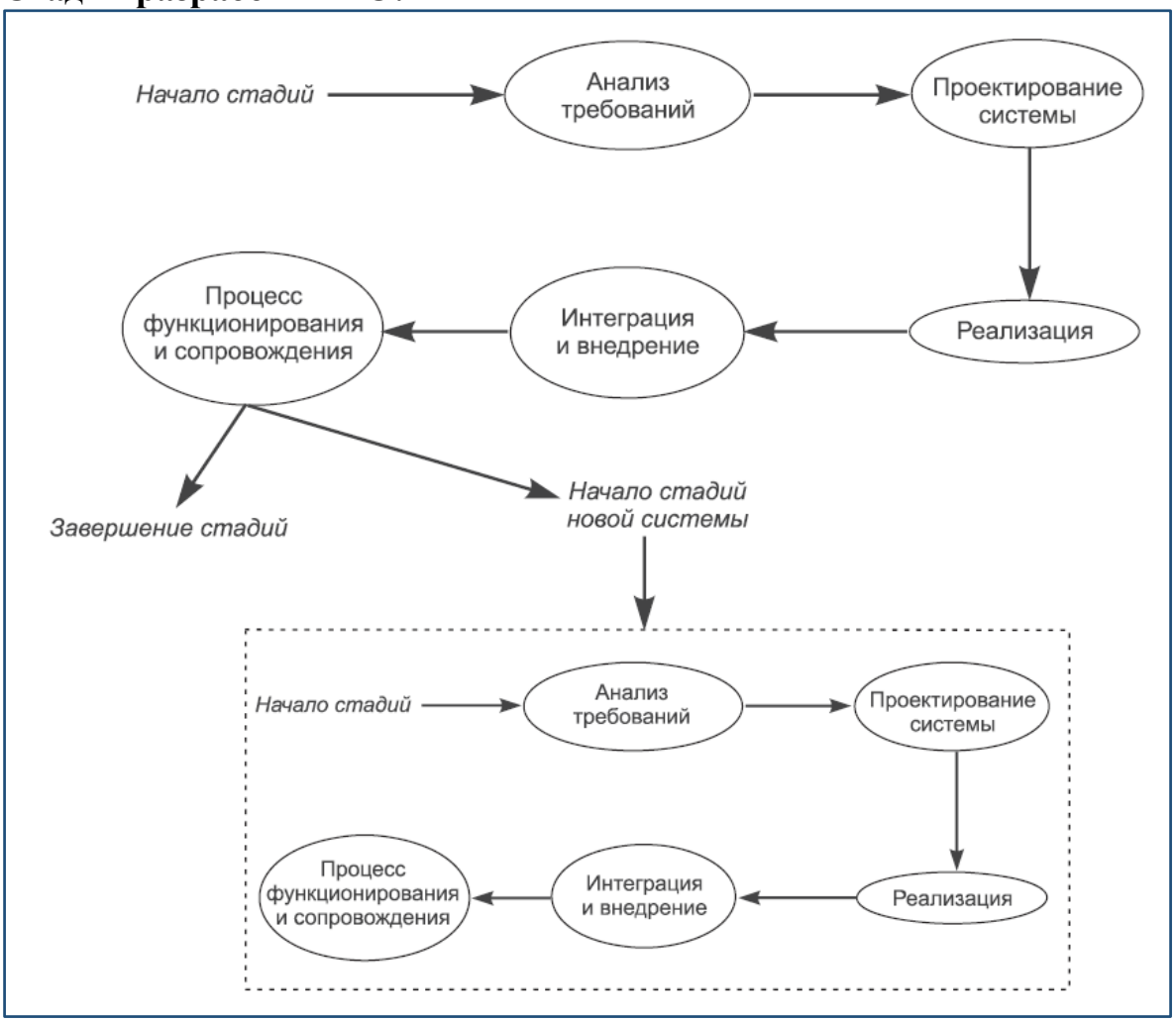
1. Жизненный цикл разработки программного обеспечения

Общепринятая модель жизненного цикла программного обеспечения, согласно которой программные системы проходят в своем развитии два этапа:
- ✓ разработка;
- ✓ сопровождение.



Каждая стадия разбивается на ряд этапов:

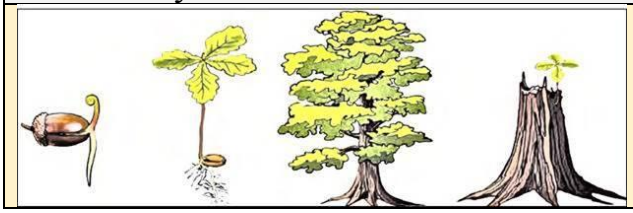
Стадии разработки ПО:



Определение:

Жизненный цикл разработки программного обеспечения –

это период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент полного его изъятия из эксплуатации.



это ряд событий, происходящих с ПО в процессе его создания и использования

Назначение

Назначение модели жизненного цикла ПО	✓ ***дает рекомендации по организации процесса разработки ПО в целом, конкретизируя его до видов деятельности, артефактов, ролей и их взаимосвязей*** ✓ ***служит основой для планирования программного проекта*** ✓ ***способствует правильному распределению обязанностей сотрудников***

МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА ПО

1. **подготовка процесса разработки;**
2. **анализ требований к системе;**
3. **проектирование системной архитектуры;**
4. **анализ требований к программным средствам;**
5. **проектирование программной архитектуры;**
6. **техническое проектирование программных средств;**
7. **программирование и тестирование программных средств;**
8. **сборка программных средств;**
9. **тестирование программных средств;**
10. **сборка системы;**
11. **тестрование системы;**
12. **внедрение;**
13. **использование и техническая поддержка.**

Модели разработки ПО: – каскадные; – итерационные; – поэтапные; – другие.	***модели отличаются по:*** – этапности (фазы, стадии, этапы); – последовательности прохождения этапов (линейная или цикличная); – гибкости (возможность подстраивать процесс под конкретные условия); – связи с определенными методологиями разработки ПО; – использованию специализированных инструментальных средств; – другие.

Стандарт ISO/IEC 12207:1995 – Information Technology – Software Life Cycle Process • принят в 1995 году ISO (International Organization for Standardization) совместно с IEC (International Electrotechnical Commission – Международная электротехническая комиссия) • в 1999г. он был принят как ГОСТ (ИСО/МЭК) 12207 – Процессы жизненного цикла программных средств (последнее издание 2010г.)
определяет организацию ЖЦ программного продукта 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.

Определения (стандарт ISO/IEC 12207):

Процесс	совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные
Каждый процесс разделен на набор действий; Каждое действие – на набор задач.	
Каждый процесс характеризуется:	✓ задачами и методами их решения; ✓ исходными данными; ✓ результатами.

Структура процессов жизненного цикла программного обеспечения:

Основные процессы	✓ приобретение ✓ поставка ✓ **разработка** ✓ эксплуатация ✓ сопровождение
Организационные процессы	✓ управление ✓ усовершенствование ✓ создание инфраструктуры ✓ обучение
Вспомогательные процессы	✓ документирование ✓ управление конфигурацией ✓ обеспечение качества ✓ верификация ✓ аттестация ✓ совместная оценка ✓ аудит ✓ разрешение проблем

Процесс разработки включает следующие действия:

- подготовительную работу;
- анализ требований к системе;
- проектирование архитектуры системы;
- анализ требований к ПО;
- проектирование архитектуры ПО;
- детальное проектирование ПО;
- кодирование и тестирование ПО;
- интеграцию ПО;
- квалификационное тестирование ПО;
- интеграцию системы;
- квалификационное тестирование системы;
- установку ПО;
- приемку ПО.

2. Виды моделей жизненного цикла

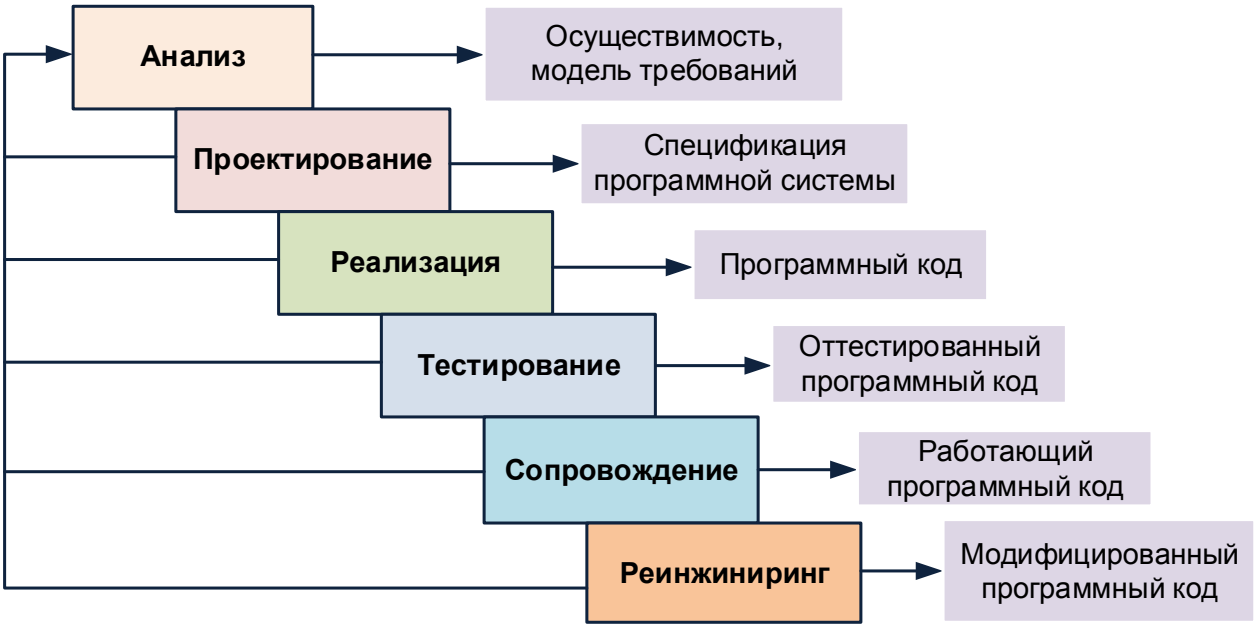
В настоящее используются следующие модели жизненного цикла:

<table>
<tr><td>Каскадная (водопадная) модель
  </td><td>– последовательное и однократное выполнение всех этапов проекта в строго фиксированном порядке;
– переход на следующий этап после полного завершения работ на предыдущем этапе.</td></tr>
<tr><td>преимущества:
– на каждой стадии формируется законченный набор проектной документации;
– выполняемые в логической последовательности стадии работ позволяют планировать сроки завершения всех работ и соответствующие затраты.</td><td>недостатки:
– выявление и устранение ошибок производится только на стадии тестирования (как следствие, неточные спецификации приводят к переработке уже принятых решений);
– реальные проекты часто требуют отклонения от стандартной последовательности шагов;
– ЖЦ основан на точной формулировке исходных требований к ПО (на практике часто случается, что в начале проекта требования заказчика определены лишь частично);
– результаты работ доступны заказчику только по завершении проекта.</td></tr>
</table>

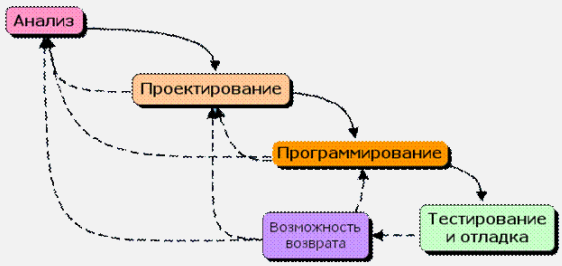
Область применения каскадной модели:

- в критически важных системах реального времени (например, управление авиационным движением или медицинским оборудованием);
- в масштабных проектах, в реализации которых задействовано несколько больших команд разработчиков;
- при разработке новой версии уже существующего продукта или переносе его на новую платформу;
- в организациях, имеющих большой практический опыт в создании программных систем определенного типа (например, бухгалтерский учет, начисление зарплаты и пр.).

Классический жизненный цикл разработки программного обеспечения:



Поэтапная модель с промежуточным контролем



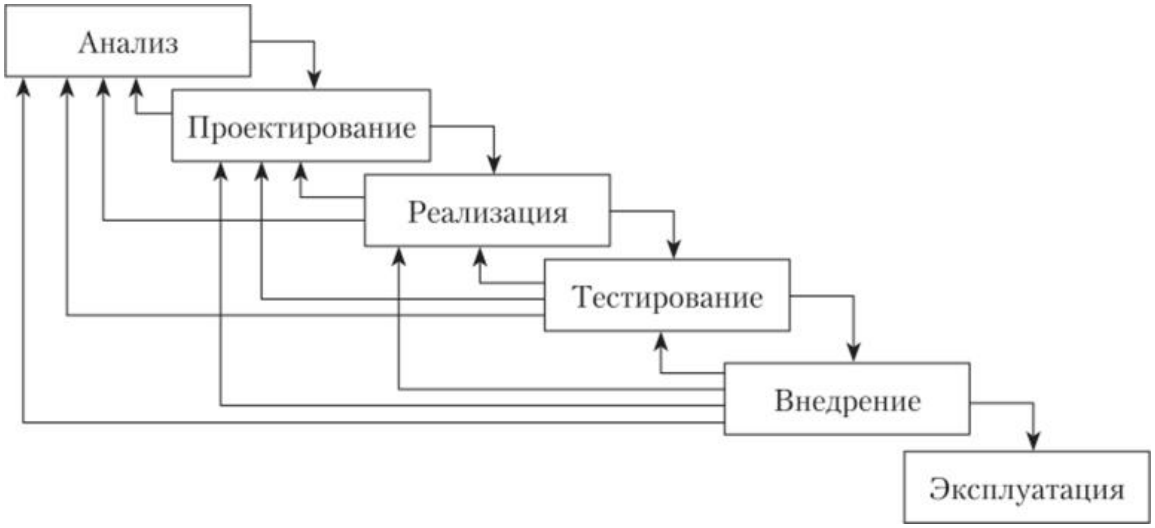
- разработка ведется итерациями с циклами обратной связи между этапами;
- корректировки между этапами учитывают существующее взаимосвязи результатов разработки на различных этапах;
- время жизни каждого из этапов растягивается на весь период разработки.

преимущества:

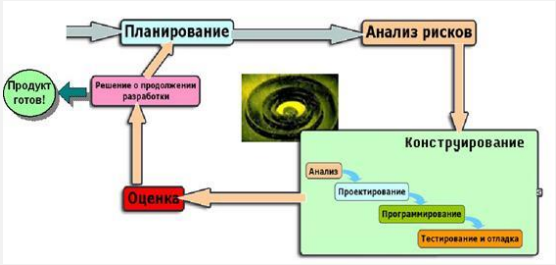
- оперативная разработка и демонстрация ПО заказчику для устранения ошибок;
- допускается отсутствие требований к ПО.

недостатки:

- сложность планирования работ и сроков завершения проекта;
- ориентирование на разработку.



Спиральная модель



– на каждом витке спирали создается очередная версия продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка.

преимущества:

– прототипирование позволяет пользователям «увидеть» систему на ранних этапах разработки;
– позволяет идентифицировать риски без особых дополнительных затрат;
– предусматривает активное участие пользователей в работах по планированию, анализу рисков, разработке и оценке полученных результатов.

недостатки:

– модель имеет усложненную структуру;
– сложность поддержания версий продукта;
– сложность оценки точки перехода на следующий цикл;
– спираль может продолжаться до бесконечности, поскольку каждая ответная реакция заказчика на созданную версию может порождать новый цикл, что отдаляет окончание работы над проектом.

Основные характеристики спиральной модели:

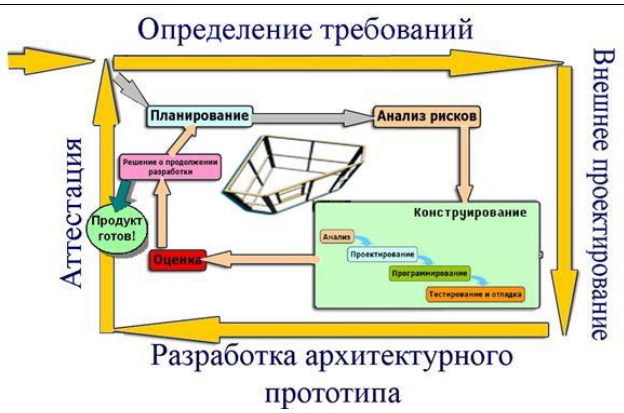
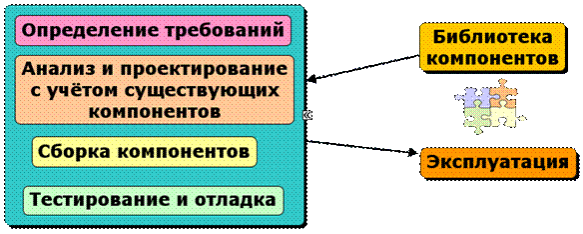
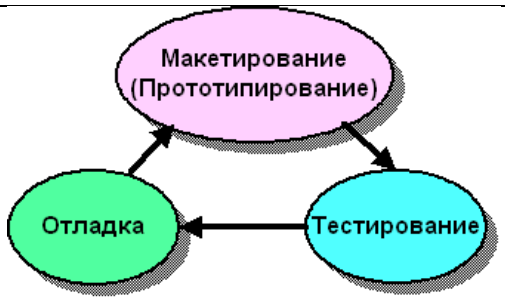
- модель состоит из последовательно следующих друг за другом этапов в пределах одного витка спирали (как и «водопадная модель»);
- внутри витка спирали этапы не имеют обратной связи; анализ результата осуществляется в конце витка и инициирует новый виток спирали;
- исправление ошибок происходит на этапе тестирования на каждом из витков спирали;
- этапы могут перекрываться во времени в пределах одного витка спирали;
- результат появляется в конце каждого витка спирали и после подробного анализа, инициируется новый виток спирали;
- при переходе от витка к витку происходит накопление и повторное использование программных средств, моделей и прототипов;
- процесс ориентирован на развитие и модификацию системы в процессе ее проектирования, на анализ рисков и издержек в процессе проектирования.

Прототип – легко поддающаяся модификации и расширению рабочая модель разрабатываемого программного средства (или системы), позволяющая пользователю получить представление о его ключевых свойствах до момента окончания полной реализации

Область применения спиральной модели

- ✓ при разработке систем, требующих большого объема вычислений (например, систем принятия решений);
- ✓ при выполнении бизнес-проектов;
- ✓ при выполнении проектов в области аэрокосмической промышленности, обороны и инжиниринга, где уже имеется позитивный опыт ее использования.

Другие модели:

каркасная модель разработки	
сборочное программирование	
исследовательское программирование	

Разные типы проектов требуют разных сочетаний подготовки и конструирования ПО. Каждый проект уникален, однако обычно проекты подпадают под общие стили разработки.

Можно выделить три самых популярных типа проектов и в большинстве случаев оптимальные методы работы над ними.

	Бизнес-системы	***Системы целевого назначения***	***Встроенные системы, от которых зависит жизнь людей***
Типичные приложения	Интернет-сайты. Сайты в интрасетях. Игры. Системы управления информацией. Системы управления информацией. Системы выплаты заработной платы. …	Встроенное ПО. Игры. Интернет-сайты. Встроенное ПО. Web-сервисы. ...	Авиационное ПО. Встроенное ПО. ПО для медицинских устройств. Операционные системы. Пакетное ПО. ...
Модели жизненного цикла	Гибкая разработка (экстремальное программирование, методология Scrum, Эволюционное прототипирование).	Поэтапная модель. Эволюционная модель. Спиральная разработка.	Поэтапная модель. Спиральная разработка. Эволюционная модель.
Внедрение приложения	Неформальная процедура внедрения	Формальная процедура внедрения.	Формальная процедура внедрения.