# 1. ВВЕДЕНИЕ В ПРОГРАММНУЮ ИНЖЕНЕРИЮ

## ЛЕКЦИЯ №1

# Основные определения

**Программные системы** состоят из совокупности программ, прошедших испытания с зафиксированными показателями, файлов конфигурации, необходимых для установки этих программ, и документации, достаточной для правильной эксплуатации этих программ.

ПС имеет все характерные черты системы:

- единая цель функционирования ПС;
- совокупность взаимосвязанных элементов;
- возможность разбивки на подсистемы, имеющие определенное самостоятельное назначение;
- иерархическая структура подсистем;
- единый критерий оценки эффективности работы программной системы.

# Основные определения

**Программа** – это любой текст на языке программирования, выполняемый на компьютере.

Программа состоит из последовательности команд (операторов), реализующих алгоритм решения конкретной задачи.

**Программный продукт (изделие)** – это программная система, подготовленная к эксплуатации (прошедшая отладку и тестирование), снабженная необходимой технической документацией, предоставляющая требуемый сервис и гарантию надежной работы, имеющая товарный знак изготовителя и код государственной регистрации.

# Основные определения

**Программный проект** (project) — это временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов.

Временный характер проекта подчеркивает, что у любого проекта есть определенное начало и завершение. Большинство проектов предпринимается для достижения устойчивого, длительного результата — время жизни конечного программного продукта может существенно превышать время жизни программного проекта. В состав программного проекта входят как люди (разработчики), так и необходимые материальные ресурсы.

# Основные определения

**Программная инженерия** (инженерия программного обеспечения) — система инженерных принципов для создания экономичного ПО, которое надежно и эффективно работает в реальных компьютерах.

Согласно международному терминологическому стандарту ISO/IEC 2382/1-93 более развернутая формулировка дается следующим образом:

**Программная инженерия** — систематическое применение научных и технологических знаний, методов и практического опыта к проектированию, реализации, тестированию и документированию программного обеспечения в целях оптимизации его производства, поддержки и качества

# Методы, средства и процессы программной инженерии

Методы обеспечивают решение широкого спектра технических задач; таких как:

- планирование и оценка программного проекта;
- анализ требований к компьютерной системе в целом и к программному обеспечению в частности;
- проектирование структур программ (и структур данных), входящих в состав ПО;
- конструирование программного текста (другие названия: кодирование, программирование, реализация);
- тестирование (выявление ошибок в созданных программах);
- сопровождение ПО, уже используемого заказчиками.

# Методы, средства и процессы программной инженерии

Средства (утилиты) программной инженерии обеспечивают автоматизированную или автоматическую поддержку методов.

В целях совместного применения утилиты могут объединяться в системы автоматизированной разработки ПО. Такие системы принято называть CASE-системами. Аббревиатура CASE расшифровывается как Computer Aided Software Engineering (программная инженерия с компьютерной поддержкой).

# Методы, средства и процессы программной инженерии

Процессы являются связующим звеном методов и утилит, они обеспечивают непрерывную технологическую цепочку разработки. Процессы определяют:

- порядок применения методов и утилит;
- формирование отчетов, форм по соответствующим требованиям;
- контроль, который помогает обеспечивать качество и координировать изменения;
- формирование контрольных точек получения результатов, по которым руководители оценивают прогресс.

# 2. Модели процессы разработки ПО

Лекция №2

# Модели процессы разработки ПО

Современная программная инженерия обеспечивает представительный набор моделей процессов, каждая из моделей имеет свои достоинства и недостатки.

Применение моделей процессов гарантирует систематический, упорядоченный подход к промышленной разработке, использованию и сопровождению ПО. Фактически эти модели вносят в программные проекты организующее инженерное начало, необходимость которого трудно переоценить.

# Виды основной деятельности базиса процессов для программной инженерии

Модели процессов состоят из таких строительных элементов, как виды деятельностей, действия и задачи.

**Деятельность** – самый крупный элемент, который ориентирован на достижение весомой цели и применяется независимо от прикладной области, размера проекта, сложности затрат или степени строгости использования инструментария программной инженерии. Деятельность состоит из действий.

**Действие** – средний элемент, охватывает набор задач, которые производят этапный рабочий продукт (например, модель результатов проектирования).

**Задача** – самый мелкий элемент. Задача фокусируется на маленькой, но хорошо определенной цели (например, на проведении тестирования модуля), которая приводит к ощутимому реальному результату.

# Виды основной деятельности базиса процессов для программной инженерии

**Модель процесса программной инженерии** – это адаптивное руководство, позволяющее команде проекта выполнять работу, указывая или выбирая подходящий набор рабочих действий и задач. Цель — создавать ПО за приемлемое время и с достаточным качеством, удовлетворяющим тех, кто спонсирует его создание, и кто будет использовать его.

Обобщенный базис процессов для программной инженерии включает пять видов основной деятельности:

- подготовка;
- планирование;
- моделирование;
- конструирование;
- развертывание.

# Виды основной деятельности базиса процессов для программной инженерии

Эти пять видов основной деятельности могут использоваться как в ходе разработки малых, простых программ, так и при создании больших и сложных ком пьютерных систем. В каждом случае детали процесса (действия, задачи, порядок их выполнения и взаимодействия) будут меняться, но виды деятельности останутся одинаковыми.

***Подготовка.*** Подготовка заключается в тесном сотрудничестве с заказчиком и другими заинтересованными лицами для понимания цели в отношении продукта и проекта, собрать их требования к характеристикам и функциям ПО.

***Планирование.*** План определяет порядок инженерной работы. Он описывает технические задачи, которые надо выполнять, наиболее вероятные факторы риска, подстерегающие команду, требуемые ресурсы, рабочие продукты и расписание работы.

# Виды основной деятельности базиса процессов для программной инженерии

***Моделирование.*** Моделирование включает в себя два действия: анализ и проектирование. Модель анализа улучшает понимание требований к ПО, а модель проектирования показывает эскиз структуры и поведения ПО, выполняющего эти требования.

***Конструирование.*** Эта деятельность объединяет два действия: генерацию программного кода ПО (ручную или автоматическую) и тестирование, которое требуется для обнаружения ошибок в коде.

***Развертывание.*** ПО поставляется заказчику, который оценивает полученный продукт и обеспечивает обратную связь, дающую возможность улучшения продукта.

# Официальная классификация процессов программной инженерии

Классификацию процессов программной инженерии задают международный стандарт ISO/IEC 12207-2008 «Systems and software engineering — Software Life Cycle Processes» и его российский аналог ГОСТ Р ИСО/МЭК 12207-2010. В стандартах определены процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

В этих стандартах процессы привязаны к основному понятию программной инженерии — жизненному циклу программного обеспечения (ЖЦ ПО).

# Определение жизненного цикла ПС

**Жизненный цикл (ЖЦ)** отражает различные временные состояния ПС и является моделью создания и использования ПС.

**Жизненный цикл ПС** – это период времени от момента возникновения идеи о создании ПС до момента ее полного выхода из употребления.

# Модели жизненного цикла

Существуют три классические модели жизненного цикла ПС

- каскадная;
- итерационная;
- спиральная модели,

каждая из которых определяет одну из стратегий разработки ПС.

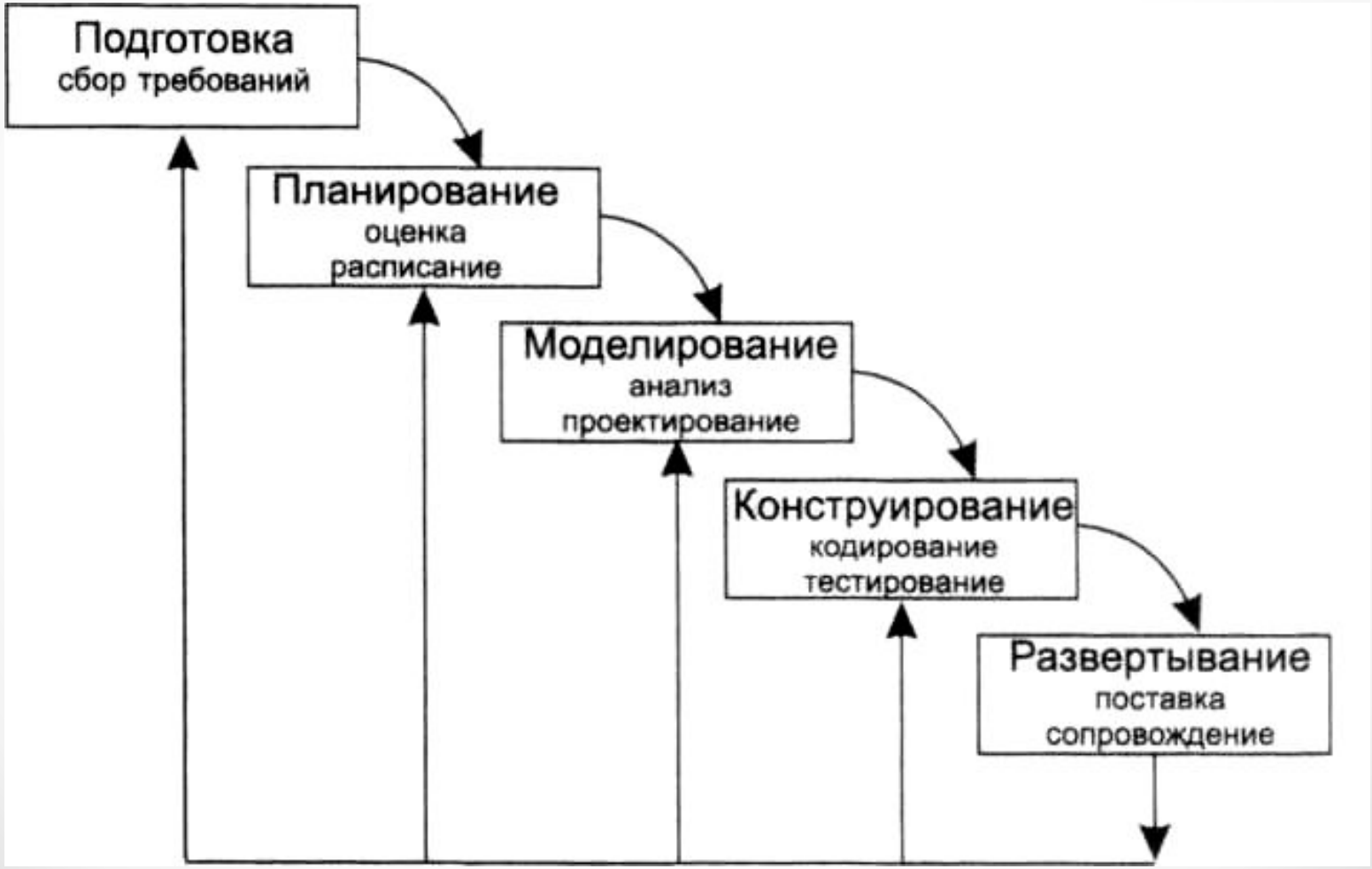
# Стратегии разработки ПО

Существуют 3 стратегии разработки ПО:

- ***однократный проход*** (водопадная стратегия) — линейная последовательность этапов разработки;
- ***инкрементная стратегия.*** В начале процесса определяются все пользовательские и системные требования, оставшаяся часть разработки 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.

# Каскадная модель разработки

**Каскадная** (или водопадная) модель подразумевает переход на следующий, иерархически нижний этап только после полного завершения работ на текущем этапе.



# Каскадная модель разработки ПС

***Подготовка*** обеспечивает активное взаимодействие с потенциальным заказчиком. Помимо оформления контракта, здесь собираются и формируются требования, определяющие характеристики и функции будущей программной системы. Этот сбор требует интенсивного диалога с заказчиком и другими заинтересованными в создании системы лицами. Фактически эти требования определяют полное задание на разработку.

***Планирование***. На этом этапе решаются задачи планирования программного проекта. В ходе планирования проекта определяются объем будущих работ и их риск, необходимые трудозатраты, формируются рабочие задачи и расписание (план-график работ).

# Каскадная модель разработки ПС

***Моделирование*** посвящено выполнению двух действий — анализу требований и проектированию. Результаты этих действий — модели — обычно записываются на графическом языке моделирования, языке картинок.

***Анализ требований*** обрабатывает набор требований к ПО, сформированный на этапе подготовки. Уточняются и детализируются функции, характеристики и интерфейс ПО. Все текстовые определения и модели документируются в спецификации анализа (техническом задании).

# Каскадная модель разработки ПС

***Проектирование*** состоит в создании представлений:

- архитектуры ПО (программы и их назначение, взаимодействие между программами);
- структурной и поведенческой организации частей архитектуры ПО (алгоритмы работы каждой программы и структуры данных);
- входного и выходного интерфейса частей архитектуры (входных и выходных форм данных).

# Каскадная модель разработки ПС

*Архитектура ПО* определяет организационную структуру программной системы, задает ее разбиение на части, связи между этими частями, механизмы взаимодействия и основные руководящие принципы для детализации дальнейшего проектирования системы.

Исходные данные для проектирования содержатся в ***спецификации анализа***. По сути, в ходе проектирования выполняется трансляция требований к ПО во множество проектных представлений. При решении задач проектирования основное внимание уделяется качеству будущего программного продукта.

# Каскадная модель разработки ПС

***Конструирование*** — этот этап включает в себя действия кодирования и тестирования.

***Кодирование***, иначе называемое программированием или реализацией, — состоит в переводе результатов проектирования в текст на языке программирования.

***Тестирование*** — выполнение программы для выявления ошибок в функциях, логике и форме реализации программного продукта. Если ошибки выявлены, запускается отладка, цель которой — устранить ошибки.

# Каскадная модель разработки ПС

***Развертывание*** — последний этап классического жизненного цикла нацелен на два действия: *поставку* разработанного продукта заказчику и сопровождение процесса эксплуатации этого продукта. Данный этап является самым длительным по времени этапом жизненного цикла ПС.

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

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

# Каскадная модель разработки

**Достоинства** каскадной модели:

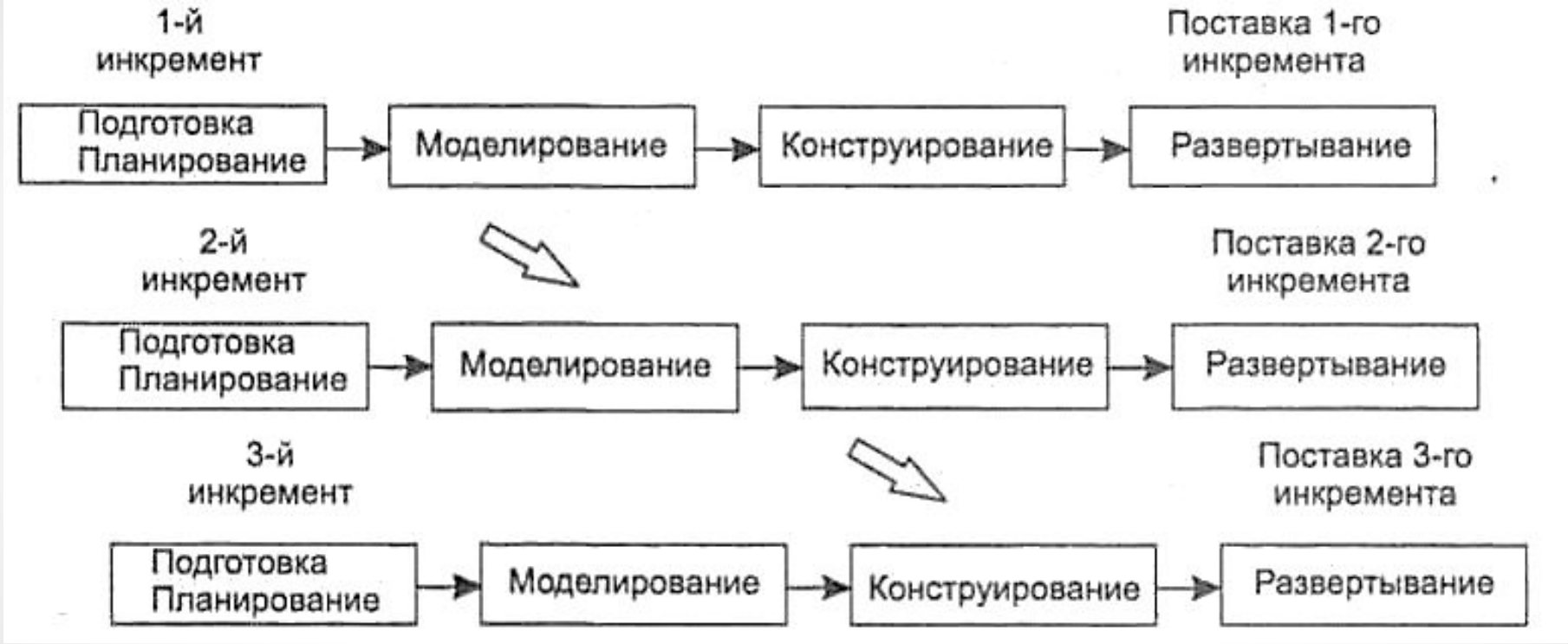
- дает план и временной график по всем этапам проекта,
- упорядочивает ход разработки.

**Недостатки** каскадной модели :

- реальные проекты часто требуют отклонения от стандартной последовательности шагов;
- цикл основан на точной формулировке исходных требований к ПО (реально в начале проекта требования заказчика определены лишь частично);
- результаты проекта доступны заказчику только в конце работы.

Здесь применяется стратегия ***однократного прохода***, когда в самом начале известны все требования, отсутствуют циклы этапов, промежуточные варианты ПС не распространяются.

# Инкрементная модель разработки ПС




Инкрементная модель является классическим примером инкрементной стратегии разработки.

# Инкрементная модель разработки ПС

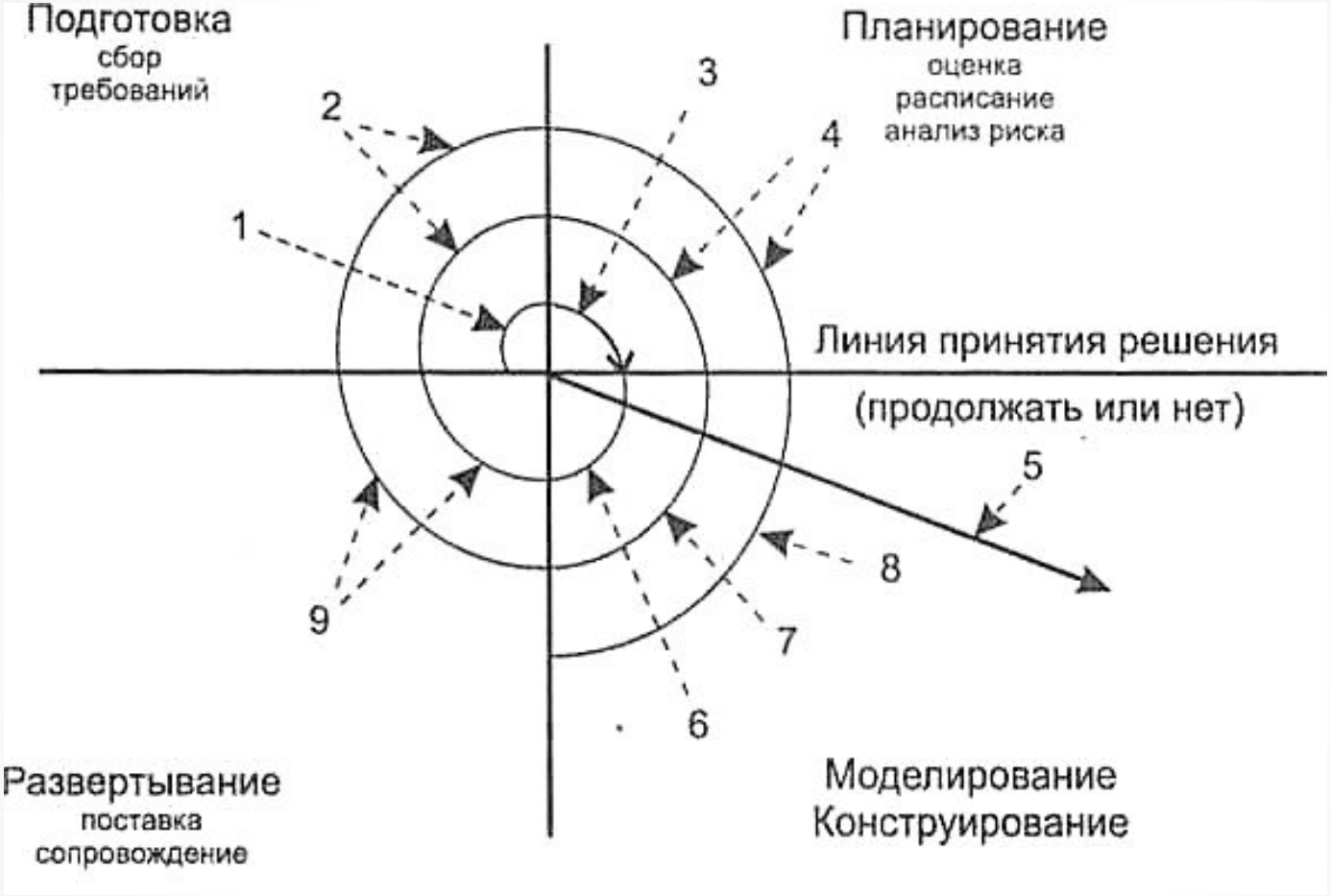
Каждая линейная последовательность здесь вырабатывает поставляемый инкремент (версию) ПО.

Первый инкремент приводит к получению базового продукта, реализующего базовые требования. План следующего инкремента предусматривает модификацию базового продукта, обеспечивающего дополнительные характеристики и функциональность.

По своей природе инкрементный процесс итеративен, инкрементная модель обеспечивает на каждом инкременте работающий продукт.

Если в процессе разработки системы меняются начальные требования, то инкрементная модель становится неэффективной

# Спиральная модель




**Спиральная модель** - классический пример применения эволюционной стратегии разработки.

# Спиральная модель

(1) начальный сбор требований проекта;
(2) начальный сбор требований проекта, но на основе рекомендаций заказчика;
(3) планирование проекта и анализ риска на основе начальных требований;
(4) планирование и анализ риска на основе реакции заказчика;
(5) переход к комплексной системе;
(6) начальный макет системы;
(7) версия системы следующего уровня;
(8) разработанная система;
(9) оценивание заказчиком.

# Спиральная модель

Модель определяет четыре действия, представляемые четырьмя квадрантами спирали.

**Подготовка** — сбор требований и ограничений.

**Планирование** — формирование плана проекта и анализ риска.

**Моделирование** и конструирование — подготовка моделей и реализация продукта следующего уровня.

**Развертывание** — оценка заказчиком текущей версии продукта.

Разработка здесь отображается движением по разворачивающейся спирали (по часовой стрелке), причем проект стартует в первом квадранте. Интегриру ющий аспект спиральной модели очевиден при учете радиального измерения спирали.

# Спиральная модель

***Достоинства спиральной модели:***

- наиболее реально (в виде эволюции) отображает разработку программного обеспечения;
- позволяет явно учитывать риск на каждом витке эволюции разработки;
- включает возможность оценки системы в итерационную структуру разработки;
- использует моделирование для уменьшения риска и совершенствования программного изделия.

***Недостатки спиральной модели:***

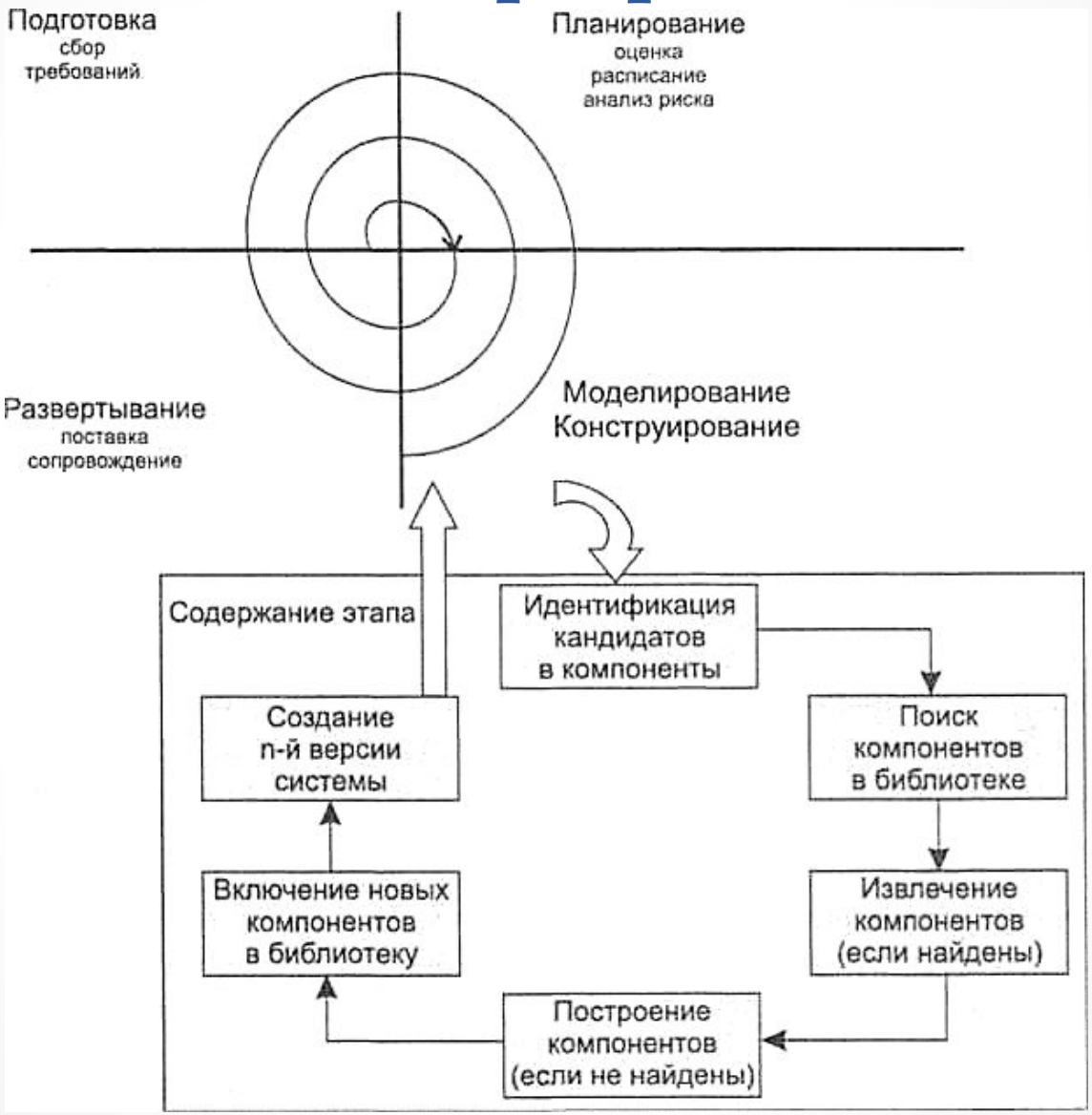
- повышенные требования к заказчику;
- трудности контроля и управления временем разработки.

# Компонентно-ориентированная модель

Компонентно-ориентированная модель является развитием спиральной модели и тоже основывается на эволюционной стратегии разработки.

В этой модели модифицируется содержание квадранта моделирования-конструирования — оно отражает тот факт, что в современных условиях новая разработка должна основываться на повторном использовании существующих программных компонентов.

# Компонентно-ориентированная методика разработки



# Объектно-ориентированная модель (Rational Objectory Process)

Фирма Rational Software, разработавшая язык UML (унифицированный язык моделирования), предложила свою модель ЖЦ, которая называется Rational Objectory Process (ROP).

Основные свойства ROP-технологии следующие:

- ROP – итеративный процесс, в течении которого происходит последовательное уточнение результатов;
- ROP направлен на создание моделей, а не на разработку каких-либо других элементов проекта (например, текстовых документов);
- действия ROP определяются в первую очередь блоками использования (use case).

# Объектно-ориентированная модель (Rational Objectory Process)

ROP разбит на циклы, каждый из которых, в свою очередь, состоит из четырех стадий:

- начальная стадия (Inception);
- разработка (Elaboration);
- конструирование (Construction);
- ввод и эксплуатация (Transition).

Результатом работы каждого такого цикла является своя версия программной системы.

Каждый цикл завершается в четко определенной точке (milestone). В этот момент времени должны достигаться определенные важные результаты и приниматься решения о продолжении дальнейшей разработки.

# Объектно-ориентированная модель (Rational Objectory Process)

**Начальная стадия** может принимать множество разных форм. Для крупных проектов – это всестороннее изучение всех возможностей реализации на протяжении нескольких месяцев. Здесь же разрабатывается бизнес-план проекта. Определяется его стоимость, примерный доход, ограничения на имеющиеся ресурсы. Другими словами, выполняется некоторый начальный анализ оценки проекта.

Окончанием начальной стадии могут служить следующие результаты:

- начальный проектный словарь терминов;
- общее описание системы (основные требования к проекту, его характеристики и ограничения);
- начальная модель вариантов использования;
- начальный бизнес-план;
- план проекта, отражающий стадии и итерации;
- один или несколько прототипов.

# Объектно-ориентированная модель (Rational Objectory Process)

На **стадии разработки** выявляются более детальные требования к системе, выполняется детальный анализ предметной области и проектирование базовой архитектуры системы, создается план конструирования и устраняются наиболее рискованные элементы проекта.

Самым важным результатом стадии разработки является описание базовой архитектуры будущей системы. Эта архитектура включает:

- модель предметной области, которая служит основой для формирования основных абстракций предметной области;
- технологическая платформа, определяющая основные элементы технологии реализации системы и взаимодействие ее элементов.

Стадия разработки занимает примерно пятую часть времени создания ПС.

На стадии разработки также оценивают время реализации каждого варианта проекта, идентифицируют наиболее серьезные риски и определяют возможности их ликвидации.

# Объектно-ориентированная модель (Rational Objectory Process)

Сущность **стадии конструирования** заключается в кодировании и тестировании разрабатываемой ПС. Для каждого цикла разработки ПС на этапе конструирования определяют последовательности итераций конструирования и вариантов использования, реализуемых на каждой итерации, которые являются одновременно инкрементными и повторяющимися.

Итерации являются инкрементными в соответствии с выполняемой функцией. Каждая итерация добавляет очередные конструкции к вариантам использования, реализованным во время предыдущих итераций.

Итерации являются повторяющимися по отношению к разрабатываемому коду. На каждой итерации некоторая часть существующего кода переписывается с целью сделать его более гибким.

Результатом стадии конструирования является продукт, готовый к передаче пользователям и содержащий, как правило, руководство пользователя и готовый к интеграции на требуемых платформах.

# Объектно-ориентированная модель (Rational Objectory Process)

Назначением **стадии ввода в эксплуатацию** является передача готового продукта в полное распоряжение конечных пользователей. Данная стадия включает

- бета-тестирование, позволяющее убедиться, что новая ПС соответствует ожиданиям пользователей;
- параллельное функционирование с существующей системой, которая подлежит постепенной замене;
- оптимизацию производительности;
- обучение пользователей и специалистов службы сопровождения.

Современные технологии программирования базируются на изложенных моделях ЖЦ ПС.

# Изменение жизненного цикла программного обеспечения при ис пользовании CASE-технологий

***CASE-технологии*** представляют собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных программных систем, основанных как на структурном, так и на объектном подходах, которые поддерживаются комплексом взаимосвязанных средств автоматизации. В основе любой CASE-технологии лежит парадигма методология/метод/нотация/средство.

***Методология*** строится на базе некоторого подхода и определяет шаги работы, их последовательность, а также правила распределения и назначения методов. ***Метод*** определяет способ достижения той или иной цели - выполнение шага работы.

***Нотацией*** называют систему обозначений, используемых для описания некоторого класса моделей. Нотации бывают графические (предоставление моделей в виде графов, диаграмм, таблиц, схем и т. п.) и текстовые (описания моделей на формальных и естественных языках). В CASE-технологиях нотации используют для описания структуры проектируемой системы, элементов данных, этапов обработки и т. п.

# CASE-технологии

***Средства*** - инструментарий для поддержки методов: средства создания и редактирования графического проекта, организации проекта в виде иерархии уровней абстракции, а также проверки соответствия компонентов разных уровней.

Различают:

CASE-средства анализа требований, проектирования спецификаций и структуры, редактирования интерфейсов (первое поколение CASE-I);

CASE-средства генерации исходных текстов и реализации интегрированного окружения поддержки полного жизненного цикла разработки программного обеспечения (второе поколение CASE-II).

# CASE-технологии

CASE-I в основном включают средства для поддержки графических моделей, проектирования спецификаций, экранных редакторов и словарей данных. CASE-II отличается существенно большими возможностями, обеспечивая: контроль, анализ и связывание системной информации и информации по управлению процессом проектирования, построение прототипов и моделей системы, тестирование, верификацию и анализ сгенерированных программ.

Автоматизируя трудоемкие операции, современные CASE-средства существенно повышают производительность труда программистов и улучшают качество создаваемого программного обеспечения:

- обеспечивают автоматизированный контроль совместимости спецификаций проекта;
- уменьшают время создания прототипа системы;
- ускоряют процесс проектирования и разработки;
- автоматизируют формирование проектной документации для всех этапов жизненного цикла в соответствии с современными стандартами;
- *частично* генерируют коды программ для различных платформ разработки;
- поддерживают технологии повторного использования компонентов системы;
- обеспечивают возможность восстановления проектной документации по имеющимся исходным кодам.

# Ускорение разработки программного обеспечения. Технология RAD

Современные технологии проектирования, разработки и сопровождения программного обеспечения должна отвечать следующим требованиям:

- поддержка полного жизненного цикла программного обеспечения;
- гарантированное достижение целей разработки с заданным качеством и в установленное время;
- возможность выполнения крупных проектов в виде подсистем, разрабатываемых группами исполнителей ограниченной численности (3-7 человек) с последующей интеграцией составных частей, и координации ведения общего проекта;
- минимальное время получения работоспособной системы;
- возможность управления конфигурацией проекта, ведения версий проекта и автоматического выпуска проектной документации по каждой версии;
- независимость выполняемых проектных решений от средств реализации (СУБД, операционных систем, языков и систем программирования);
- поддержка комплексом согласованных CASE-средств, обеспечивающих автоматизацию процессов, выполняемых на всех стадиях жизненного цикла.

# Быстрая разработка приложений

Этим требованиям отвечает *технология RAD* (Rapid Application Development - **Быстрая разработка приложений**), которая использует инкрементную стратегию разработки при малых сроках разработки в 2 -3 месяца, основывается на всевозможных средствах автоматизации и компонентно-ориентированной технологии.

Очень часто RAD используются при разработке информационных систем. Это вполне понятно, т.к. существует целый ряд распространенных программных продуктов (ERWIN, BPWIN, Rational Rose и др.), автоматизирующих проектирование баз данных, являющихся основой информационных систем.

**Жизненный цикл ПС по методологии RAD** состоит из четырех фаз:

- анализа и планирования требований;
- проектирования;
- построения;
- внедрения.

# Быстрая разработка приложений

На **фазе проектирования** осуществляют следующие действия:

- детализируют реализуемые задачи; при необходимости для каждого элементарного процесса создают частичный прототип: экран, диалог, отчет, устраняющий неясности или неоднозначности; определяются требования разграничения доступа к данным;
- после детального определения состава процессов оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении ее на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время, т.е. порядка 60 - 90 дней;
- привлекают пользователей к участию в техническом проектировании системы под руководством специалистов-разработчиков;
- применяют CASE-средства для быстрого получения работающих прототипов приложений;
- используют пользователей для тестирования варианта приложения, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе;
- определяют набор необходимой документации.

# Быстрая разработка приложений

**Результатом данной фазы** должны быть:

- общая информационная модель системы;
- функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;
- точно определенные с помощью CASE-средств интерфейсы между автономно разрабатываемыми подсистемами;
- построенные прототипы экранов, отчетов, диалогов.

Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы.

# Быстрая разработка приложений

На **фазе построения** выполняется непосредственно сама быстрая разработка приложения.

На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей. Программный код частично формируется при помощи автоматических генераторов, получающих информацию непосредственно из репозитория CASE-средств.

Конечные пользователи на этой фазе оценивают получаемые результаты и вносят изменения, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется непосредственно в процессе разработки

# Быстрая разработка приложений

После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения с остальными, а затем тестирование системы в целом.

Завершается физическое проектирование системы:

- определяется необходимость распределения данных;
- производится анализ использования данных;
- производится физическое проектирование базы данных;
- определяются требования к аппаратным ресурсам;
- определяются способы увеличения производительности;
- завершается разработка документации проекта.

**Результатом фазы** является готовая система, удовлетворяющая всем согласованным требованиям.

# Быстрая разработка приложений

На **фазе внедрения** производятся:

- обучение пользователей,
- организационные изменения в текущей работе организации или фирмы параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой).

Так как фаза внедрения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, как правило, на этапе проектирования системы.

# Быстрая разработка приложений

**Методология RAD неприменима** для построения**:**

- сложных расчетных программ;
- операционных систем;
- программ управления техническими объектами в реальном режиме времени, обеспечивающих высокую степень надежности функционирования объектов;
- для ПС, в которых отсутствует ярко выраженная интерфейсная часть