БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ШУЛЬДОВА СВЕТЛАНА ГЕОРГИЕВНА,

к.т.н., доцент

svetashuldova@gmail.com

Ауд. 212-4

СТРУКТУРА ДИСЦИПЛИНЫ

- Лекций – 44 часа (22 лекции)
- Лабораторных занятий – 16 часов (4 лабораторные работы)
- Зачет

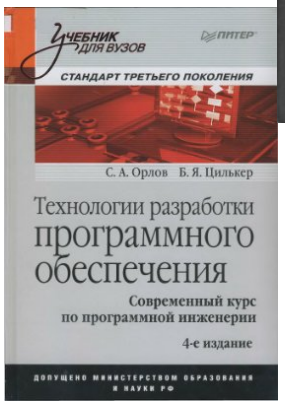
СТРУКТУРА ДИСЦИПЛИНЫ

Контрольные срезы по разделам дисциплины:

1. Стратегии разработки программного обеспечения (ПО), модели жизненного цикла ПО – **10/03/2020**
2. Классические методологии разработки программных средств (ПС) – **05/05/2020**
3. Основы объектно-ориентированного анализа и проектирования ПС. Инструментальные средства разработки - **04/06/2020**

ЛИТЕРАТУРА (ОСНОВНАЯ)

1. *Бахтизин, В. В. Технология разработки программного обеспечения : учеб. пособие / В В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2010.*
2. *Брауде Э. Технология разработки программного обеспечения* - СПб.: Питер, 2004.
3. *Орлов, С. А. Технологии разработки программного обеспечения / С. А. Орлов. – СПб. : Питер, 2012.*

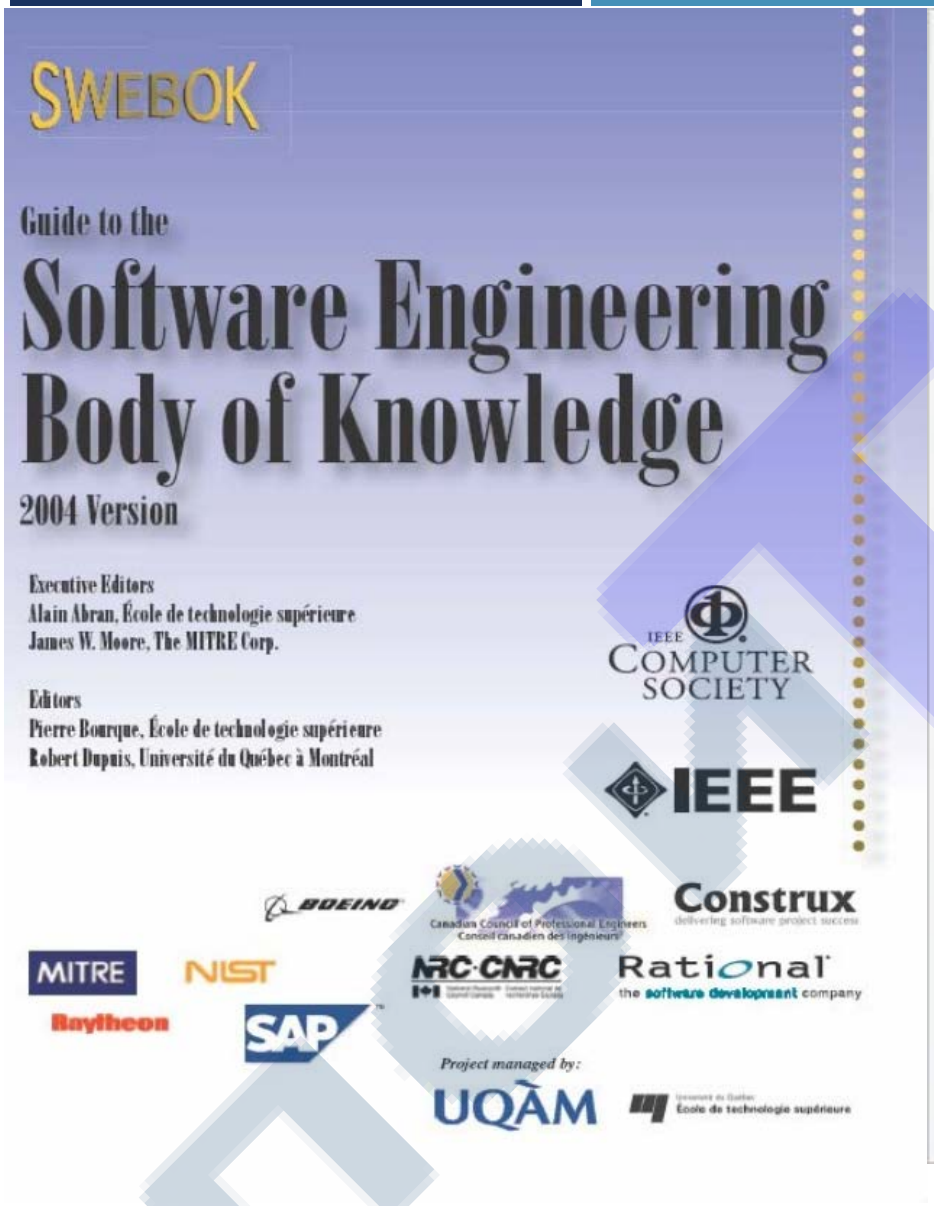


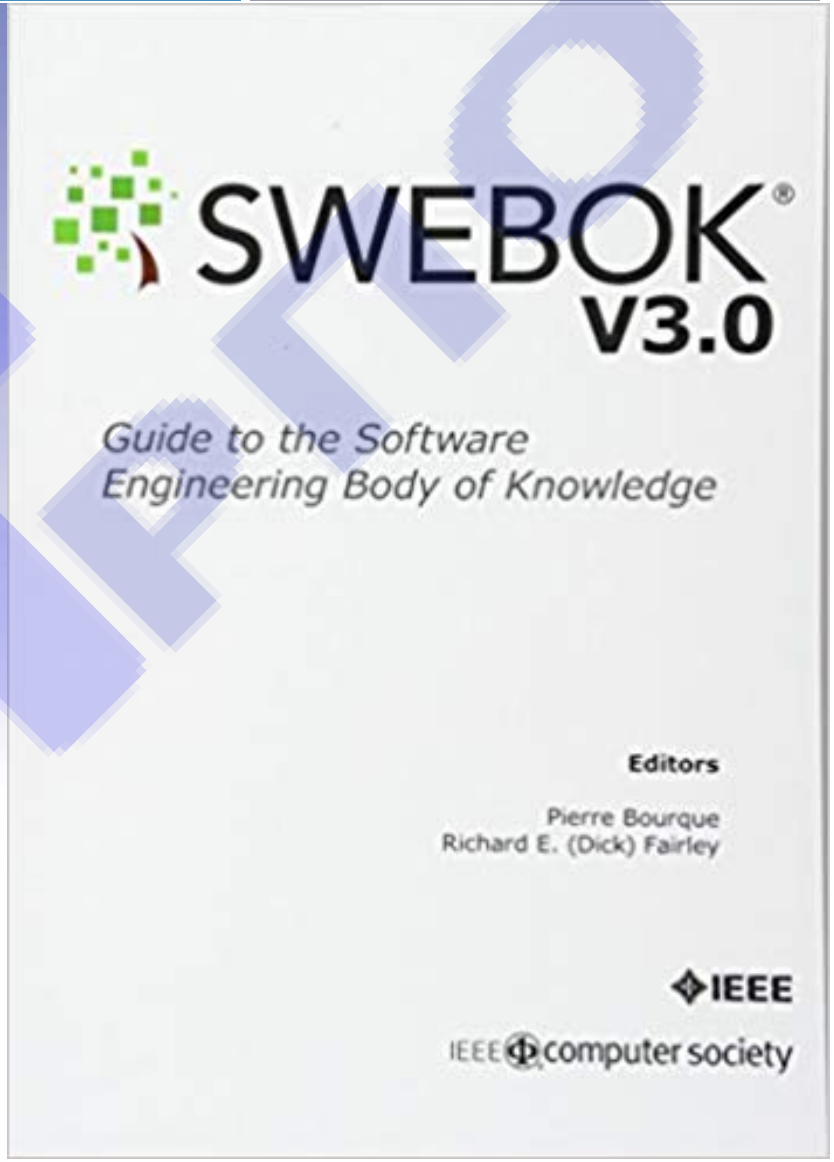
SWEBOK

The Guide to the Software Engineering Body of Knowledge, SWEBOK

Руководство к Своду Знаний по программной инженерии

- IEEE - Computer Society of the Institute for Electrical and Electronic Engineers, IEEE Computer Society – IEEE-CS (Компьютерное Общество)
- http://www.ieee.org


SWEBOK
Guide to the
Software Engineering
Body of Knowledge
2004 Version
Executive Editors
Alain Abran, École de technologie supérieure
James W. Moore, The MITRE Corp.
Editors
Pierre Bourque, École de technologie supérieure
Robert Dupuis, Université du Québec à Montréal
IEEE COMPUTER SOCIETY
IEEE
BOEING
Canadian Council of Professional Engineers
Conseil canadien des ingénieurs
Construx
delivering software project success
MITRE
NIST
NRC·CNRC
Rational
the software development company
Raytheon
SAP
Project managed by:
UQÀM
École de technologie supérieure


SWEBOK®
V3.0
Guide to the Software
Engineering Body of Knowledge
Editors
Pierre Bourque
Richard E. (Dick) Fairley
IEEE
IEEE computer society

ВОПРОСЫ:

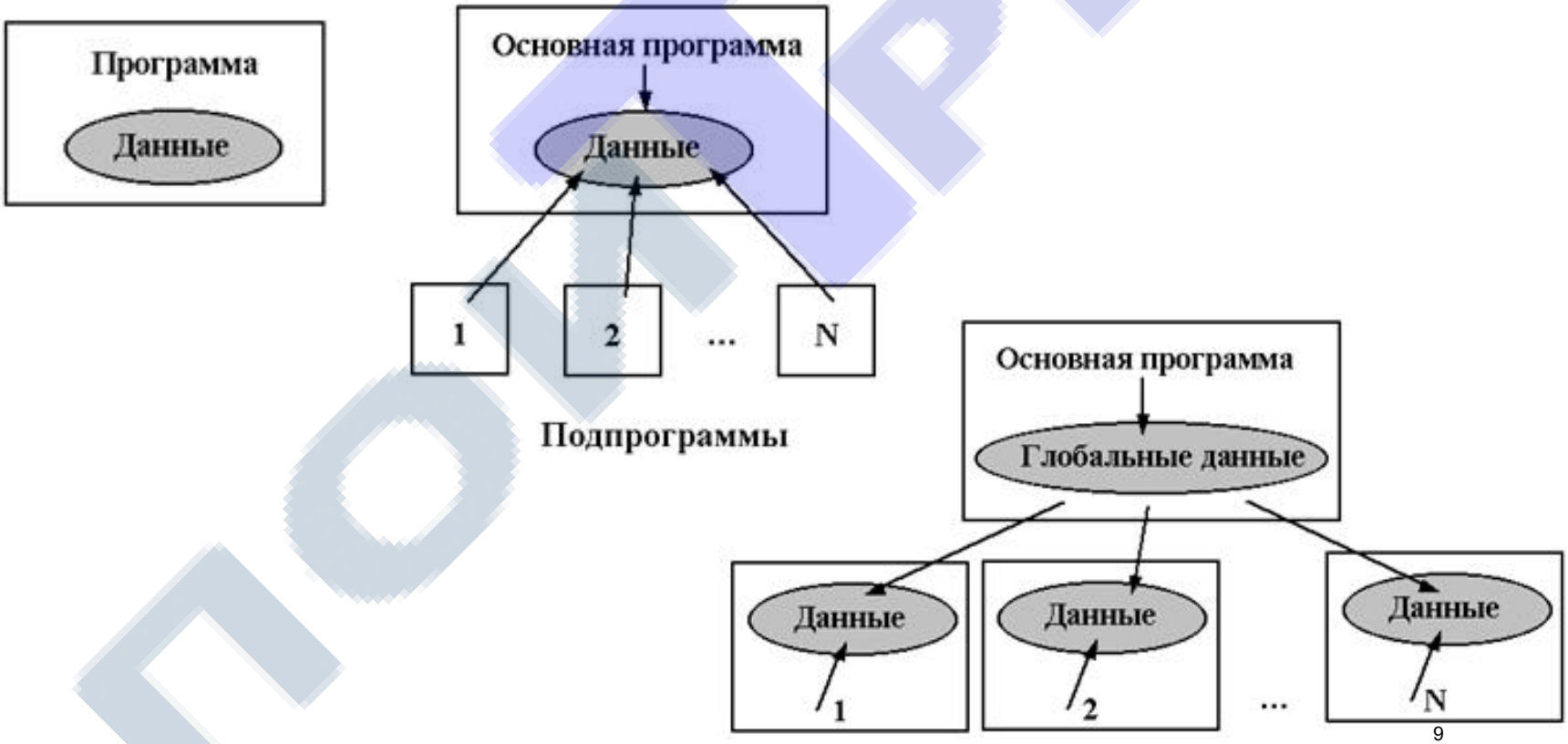
1. Этапы развития технологии разработки программного обеспечения (ПО)
2. Основные понятия и определения
3. Жизненный цикл ПО. Процессы ЖЦ ПО
4. Стратегии разработки ПО

ЭТАПЫ РАЗВИТИЯ

1. **«Стихийное» программирование** (до середины 60-х годов XX в.)
2. **Структурный подход к программированию** (60 – 70-е годы XX в.)
3. **Объектный подход к программированию** (с середины 80-х до конца 90-х годов XX в.)
4. **Компонентный подход и CASE-технологии** (с середины 90-х годов XX в.)

ЭТАПЫ РАЗВИТИЯ: «СТИХИЙНОЕ» ПРОГРАММИРОВАНИЕ

Первый этап – «стихийное» программирование



ЭТАПЫ РАЗВИТИЯ: «СТИХИЙНОЕ» ПРОГРАММИРОВАНИЕ

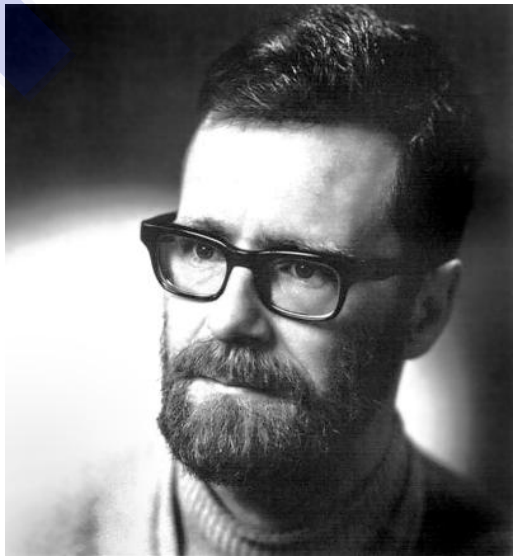
- Программирование = искусство
- Машинные языки → ассемблеры → алгоритмические языки (Fortran, Algol)
- Концепция повторного использования подпрограмм, что повысило производительность труда программиста
- Разработка "снизу вверх"
- Итог – **кризис** программирования

ЭТАПЫ РАЗВИТИЯ: СТРУКТУРНЫЙ ПОДХОД

- Нисходящее функциональное проектирование («сверху вниз»)
- Декомпозиция (разбиение на части) – представление задачи в виде иерархии подзадач с целью последующей реализации в виде отдельных небольших подпрограмм
- Метод проектирования алгоритмов – метод пошаговой детализации
- Структурное кодирование без goto

ЭТАПЫ РАЗВИТИЯ: СТРУКТУРНЫЙ ПОДХОД

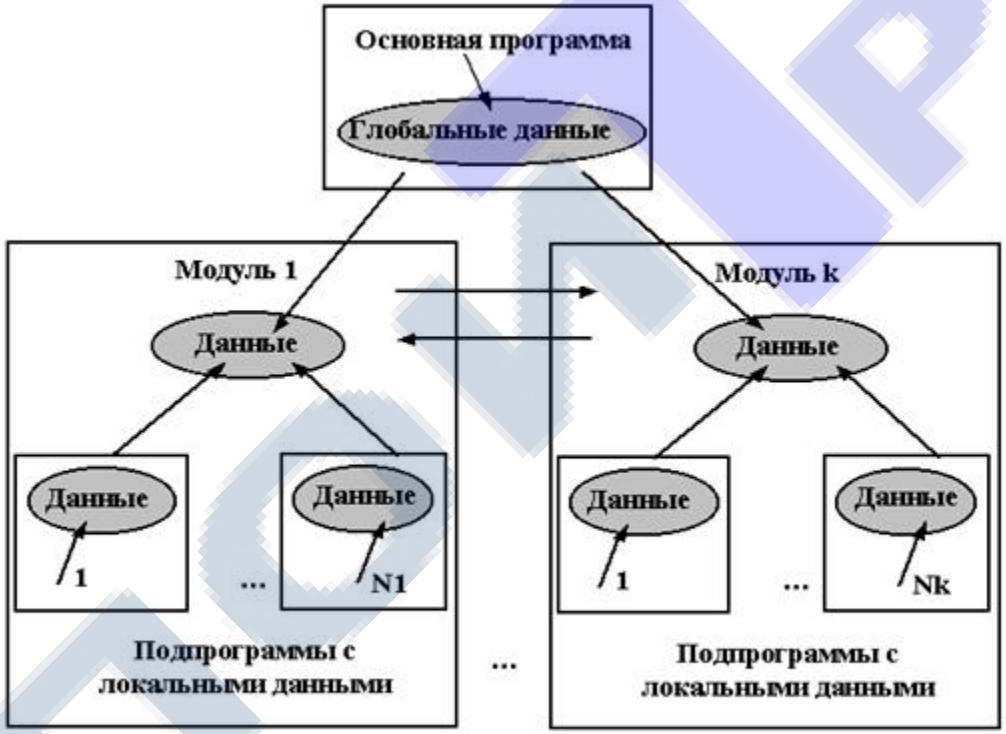
- «О вреде использования операторов GOTO» (GOTO statement considered harmful, 1968)
- Для записи любой программы в принципе достаточно только трех конструкций управления – последовательного выполнения, ветвления и цикла. То есть теоретически необходимость в использовании операторов перехода отсутствует.



Эдсгер Вибе Дейкстра

ЭТАПЫ РАЗВИТИЯ: СТРУКТУРНЫЙ ПОДХОД

Структурный подход к программированию



Модули с локальными данными и подпрограммами

- ✓ Pascal
- ✓ C

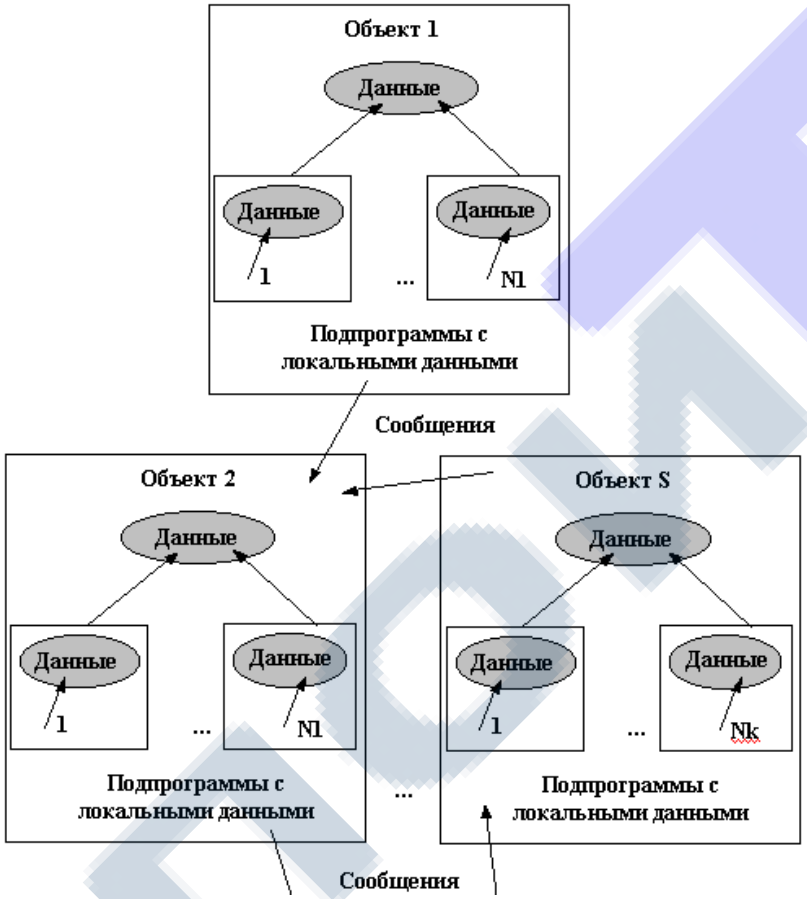
ЭТАПЫ РАЗВИТИЯ: СТРУКТУРНЫЙ ПОДХОД

- **Технология модульного программирования** – выделение групп подпрограмм, использующих одни и те же глобальные данные, в отдельно компилируемые модули.
- Структурный подход + модульное программирование = **надежные программы**, размер которых не превышает 100 000 операторов.
- **Узким местом** модульного программирования стали межмодульные интерфейсы, ошибки в которых трудно обнаружить по причине раздельной компиляции модулей (ошибки выявляются только при выполнении программы).

ЭТАПЫ РАЗВИТИЯ: ОБЪЕКТНЫЙ ПОДХОД

- **Программы** = совокупность объектов, каждый из которых является экземпляром определенного типа (класса), а классы образуют иерархию с наследованием свойств.
- Взаимодействие программных объектов осуществляется путем передачи сообщений.

ЭТАПЫ РАЗВИТИЯ: ОБЪЕКТНЫЙ ПОДХОД



- Создание сред **визуального программирования**
- Языки визуального объектно-ориентированного программирования (Delphi, Visual C++, C# и т. д.)

ЭТАПЫ РАЗВИТИЯ: ОБЪЕКТНЫЙ ПОДХОД

Недостатки

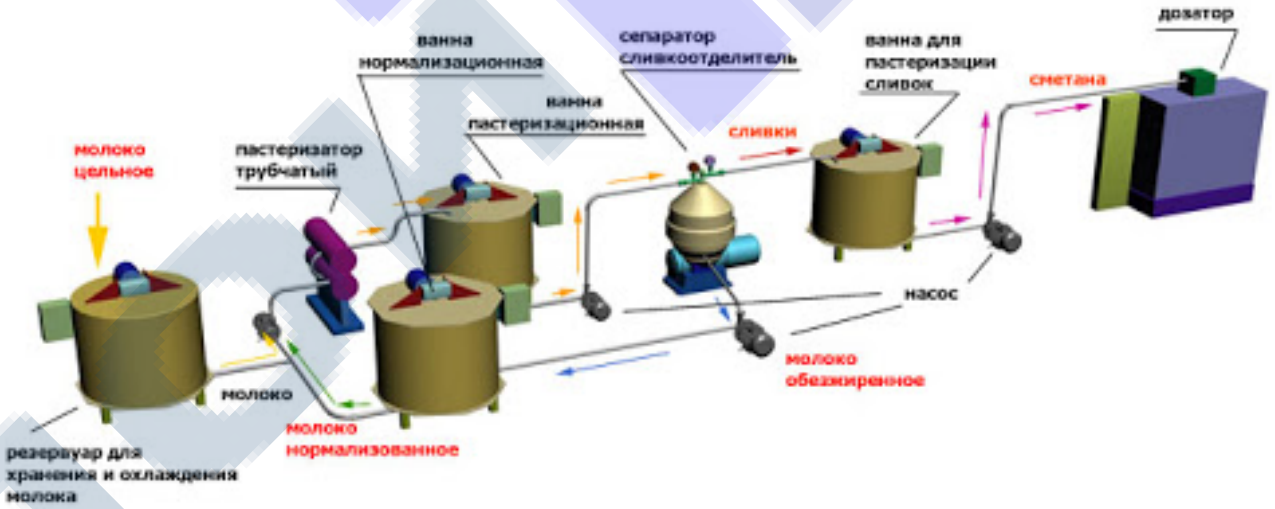
- противоречивые и переусложненные паттерны
- сложность и неоднозначность ООП-декомпозиции
- разработка программного обеспечения с использованием средств и возможностей *одного* языка программирования высокого уровня *одного* компилятора
- изменение реализации одного из программных объектов связано с перекомпоновкой всего ПО

ЭТАПЫ РАЗВИТИЯ: КОМПОНЕНТНЫЙ ПОДХОД И CASE-ТЕХНОЛОГИИ

- Построение программного обеспечения из отдельных компонентов физически отдельно существующих частей программного обеспечения, которые взаимодействуют между собой через *стандартизованные двоичные интерфейсы*.
- Объекты-компоненты можно собрать в динамически вызываемые библиотеки или исполняемые файлы, распространять в двоичном виде (без исходных текстов) и использовать в любом языке программирования, поддерживающем соответствующую технологию.

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

- **Технология** — совокупность производственных методов и процессов в определённой отрасли производства, а также научное описание способов производства.



ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

- **Технология разработки программного обеспечения** совокупность производственных процессов, приводящая созданию требуемого программного средства, а также описание этой совокупности процессов
- Продуктом технологии разработки программного обеспечения является **программа**, эффективно и надёжно выполняющая требуемые функции на реальных компьютерах.

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

- **Программное обеспечение** (software) — совокупность компьютерных программ обработки данных и необходимых для их эксплуатации документов.
- **Программное средство** – ограниченная часть программного обеспечения системы обработки информации, имеющая определенное функциональное назначение
- **Программный продукт** (software product) — совокупность компьютерных программ, процедур и связанных с ними документации и данных, предназначенная для удовлетворения потребностей пользователей, широкого распространения и продажи.

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

- В программе имеется ошибка, если она не выполняет того, что разумно ожидать от неё пользователю.

Гленфорд Дж. Майерс

- Понятие ошибки в программе является существенно не формальным

ОСОБЕННОСТИ РАЗРАБОТКИ ПО

1. Неформальный характер требований к ПС (постановка задачи) и понятия ошибки в нём, но формализованный основной объект разработки – программы ПС.
2. Разработка ПС носит творческий характер
3. ПС представляет собой некоторую совокупность текстов (т.е. статических объектов), смысл же (семантика) этих текстов выражается процессами обработки данных и действиями пользователей, запускающих эти процессы (т.е. является динамическим)
4. ПС при своём использовании (эксплуатации) не расходуется и не расходует используемых ресурсов

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

- **Технология разработки программного обеспечения** – это система инженерных принципов для создания экономичного ПО с заданными характеристиками качества в установленные сроки.
- **Методология разработки ПО** – система принципов и способов организации процесса разработки программ.
- Цель методологии разработки ПО – внедрение методов разработки программ, обеспечивающих достижение соответствующих характеристик качества.

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Метод в технологии разработки ПО - совокупность

- концепций и теоретических основ, реализующих :
  - структурный,
  - объектно-ориентированный ,
  - компонентный

подходы к разработке ПО;

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

- нотаций, используемых для построения моделей статической структуры и динамики поведения проектируемой системы;
- процедур, определяющих практическое применение метода: последовательность и правила построения моделей, критерии, используемые для оценки результатов.

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

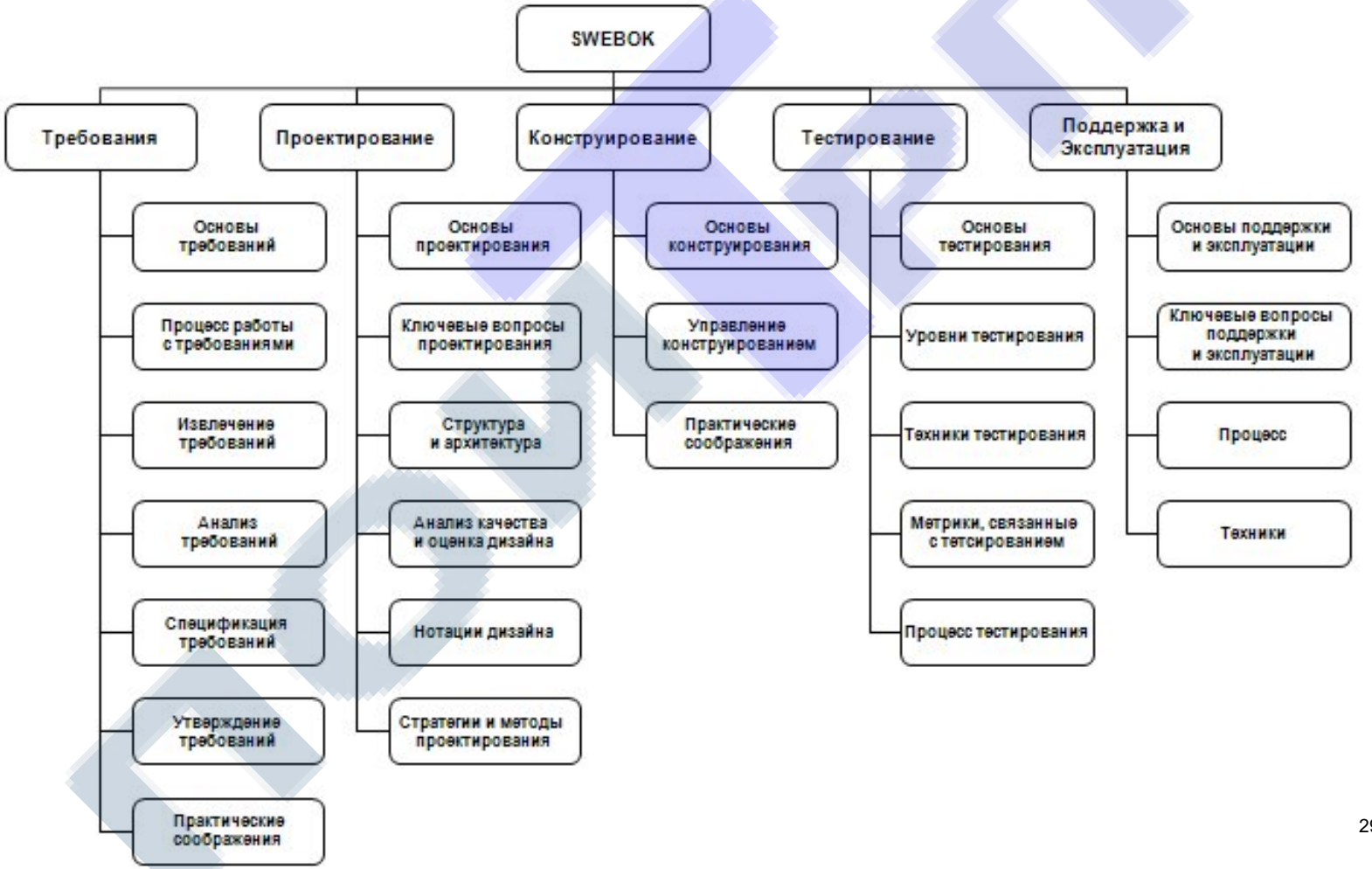
- **Инструментальные средства** – совокупность программ и программных комплексов, обеспечивающих технологию разработки, отладки и внедрения создаваемых программных продуктов.

ПРОГРАММНАЯ ИНЖЕНЕРИЯ

- 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

ПРОГРАММНАЯ ИНЖЕНЕРИЯ



СТАНДАРТЫ

Стандарт (standard) - норма, образец, мерило:

- утверждаемый компетентным органом нормативно-технический документ, устанавливающий комплекс норм и правил по отношению к объекту стандартизации
- типовой образец, эталон, модель, принимаемые за исходные для сопоставления с ними других предметов

ОСНОВНЫЕ ТИПЫ СТАНДАРТОВ

- **Корпоративные стандарты** разрабатываются крупными фирмами с целью повышения качества своей продукции. Создаются на основе собственного опыта компании, но с учетом требований мировых стандартов. Не сертифицируются, но являются обязательными для применения внутри корпорации
- **Отраслевые стандарты** действуют в пределах организаций некоторой отрасли (министерства). Разрабатываются с учетом требований мирового опыта и специфики отрасли. Подлежат сертификации

ОСНОВНЫЕ ТИПЫ СТАНДАРТОВ

- **Государственные стандарты** (ГОСТы) принимаются государственными органами и имеют силу закона. Разрабатываются с учетом мирового опыта или на основе отраслевых стандартов. Могут иметь как рекомендательный, так и обязательный характер. Для сертификации создаются государственные или лицензированные органы сертификации
- **Международные стандарты** разрабатываются специальными международными организациями на основе мирового опыта и лучших корпоративных стандартов. Имеют сугубо рекомендательный характер

РАЗРАБОТЧИКИ СТАНДАРТОВ

- **ISO** - The International Standards Organization - международная организация по стандартизации, работающая в сотрудничестве с IEC - The International Electrotechnical Commission - международной электротехнической комиссией
- **IEEE Computer Society** - профессиональное объединение специалистов в области программной инженерии
- **ACM** - Association for Computing Machinery – Ассоциация по вычислительной технике

ЖИЗНЕННЫЙ ЦИКЛ ПО

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

ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА (ISO/IEC 12207)

основные

- приобретение (заказ)
- поставка
- разработка
- эксплуатация
- сопровождение

вспомогательные

- документирование
- управление конфигурацией
- обеспечение качества
- верификация
- валидация (аттестация)
- оценка (совместный просмотр)
- аудит
- решение проблем

организационные

- управление проектами
- создание и сопровождение инфраструктуры проекта
- усовершенствование (определение, оценка и улучшение самого ЖЦ)
- обучение

ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА

Основные:

- **Приобретение** (действия и задачи заказчика, приобретающего ПО)
- **Поставка** (действия и задачи поставщика, который снабжает заказчика программным продуктом или услугой)
- **Разработка** (действия и задачи, выполняемые разработчиком: создание ПО, оформление проектной и эксплуатационной документации, подготовка тестовых и учебных материалов и т. д.)
- **Эксплуатация** и **Сопровождение** — внесений изменений в ПО в целях исправления ошибок, повышения производительности или адаптации к изменившимся условиям работы или требованиям.

ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА

Вспомогательные

- Документирование (формализованное описание информации, созданной в течение ЖЦ ПО)
- Управление конфигурацией (применение административных и технических процедур на всем протяжении ЖЦ ПО для определения состояния компонентов ПО, управления его модификациями).
- Обеспечение качества (обеспечение гарантий того, что ИС и процессы ее ЖЦ соответствуют заданным требованиям и утвержденным планам)
- Верификация (определение того, что программные продукты, являющиеся результатами некоторого действия, полностью удовлетворяют требованиям или условиям, обусловленным предшествующими действиями)

ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА

Вспомогательные

- **Аттестация** (определение полноты соответствия заданных требований и созданной системы их конкретному функциональному назначению).
- **Совместная оценка** (оценка состояния работ по проекту: контроль планирования и управления ресурсами, персоналом, аппаратурой, инструментальными средствами)
- **Аудит** (определение соответствия требованиям, планам и условиям договора)
- **Разрешение проблем** (анализ и решение проблем, независимо от их происхождения или источника, которые обнаружены в ходе разработки, эксплуатации, сопровождения или других процессов)

ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА

Организационные

- **Управление** (действия и задачи, которые могут выполняться любой стороной, управляющей своими процессами)
- **Создание инфраструктуры** (выбор и сопровождение технологии, стандартов и инструментальных средств, выбор и установка аппаратных и программных средств, используемых для разработки, эксплуатации или сопровождения ПО)
- **Усовершенствование** (оценка, измерение, контроль и усовершенствование процессов ЖЦ)
- **Обучение** (первоначальное обучение и последующее постоянное повышение квалификации персонала

БАЗОВЫЕ СТРАТЕГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

- Каскадная
- Инкрементная
- Эволюционная

Определяются характеристиками:

- проекта
- требований к программному продукту
- команды разработчиков
- команды пользователей.

КАСКАДНАЯ СТРАТЕГИЯ РАЗРАБОТКИ

КАСКАДНАЯ СТРАТЕГИЯ (1970, УИНСТОН РОЙС)

Характеристика:

- последовательное выполнение входящих в ее состав этапов;
- окончание каждого предыдущего этапа до начала последующего;
- отсутствие (или определенным ограничением) возврата к предыдущим этапам;
- наличие результата только в после завершения всех этапов.

КАСКАДНАЯ СТРАТЕГИЯ: ДОСТОИНСТВА

1. Простота применения стратегии
2. Простота планирования, контроля и управления проектом
3. Доступность для понимания заказчиками

КАСКАДНАЯ СТРАТЕГИЯ: НЕДОСТАТКИ

1. Сложность полного формулирования требований в начале процесса разработки и невозможность их динамического изменения на протяжении ЖЦ
2. Линейность структуры процесса разработки
3. Непригодность промежуточных продуктов для использования
4. Недостаточное участие пользователя в процессе разработки ПС – невозможность предварительной оценки пользователем качества программного средства.

КАСКАДНАЯ СТРАТЕГИЯ: ОБЛАСТЬ ПРИМЕНЕНИЯ

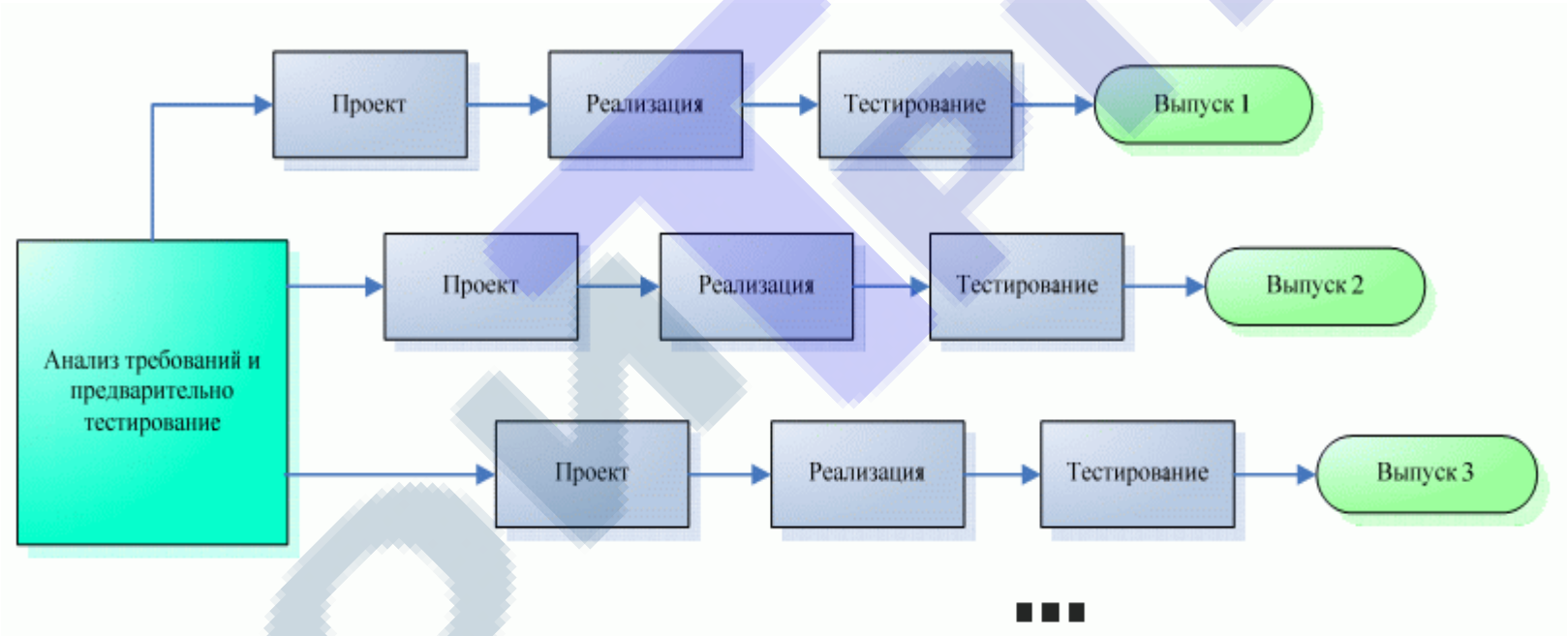
1. Проекты с четкими, неизменяемыми в течение ЖЦ требованиями и понятной реализацией.
2. Проекты невысокой сложности, например:
   - создание программного средства или системы такого же типа, как уже разрабатывались разработчиками;
   - создание новой версии уже существующего программного средства или системы;
   - перенос уже существующего продукта на новую платформу.
3. В качестве составной части моделей ЖЦ, реализующих другие стратегии разработки при выполнении больших проектов.

ИНКРЕМЕНТНАЯ СТРАТЕГИЯ: ХАРАКТЕРИСТИКА

- многократное выполнение этапов разработки с запланированным улучшением результата;
- полное определение всех требований к программному средству (системе) в начале процесса разработки;
- полный набор требований реализуется постепенно в соответствии с планом в последовательных циклах разработки.

Результат каждого цикла называется **инкрементом**.

ИНКРЕМЕНТНАЯ СТРАТЕГИЯ РАЗРАБОТКИ



ИНКРЕМЕНТНАЯ СТРАТЕГИЯ: ДОСТОИНСТВА

1. Возможность получения функционального продукта после реализации каждого инкремента
2. Короткая продолжительность создания инкремента
3. Предотвращение реализации громоздких спецификаций требований
4. Стабильность требований во время создания определенного инкремента
5. Возможность учета изменившихся требований
6. Снижение рисков по сравнению с каскадной стратегией
7. Включение в процесс пользователей, что позволяет оценить функциональные возможности продукта на более ранних этапах разработки и в конечном итоге приводит к повышению качества программного продукта, снижению затрат и времени на его разработку.

ИНКРЕМЕНТНАЯ СТРАТЕГИЯ: НЕДОСТАТКИ

1. Необходимость полного функционального определения системы или программного средства в начале ЖЦ для обеспечения планирования инкрементов и управления проектом
2. Возможность текущего изменения требований к системе или программному средству, которые уже реализованы в предыдущих инкрементах
3. Сложность планирования и распределения работ;
4. Проявление человеческого фактора, связанного с тенденцией к оттягиванию решения трудных проблем на поздние инкременты, что может нарушить график работ или снизить качество программного продукта.

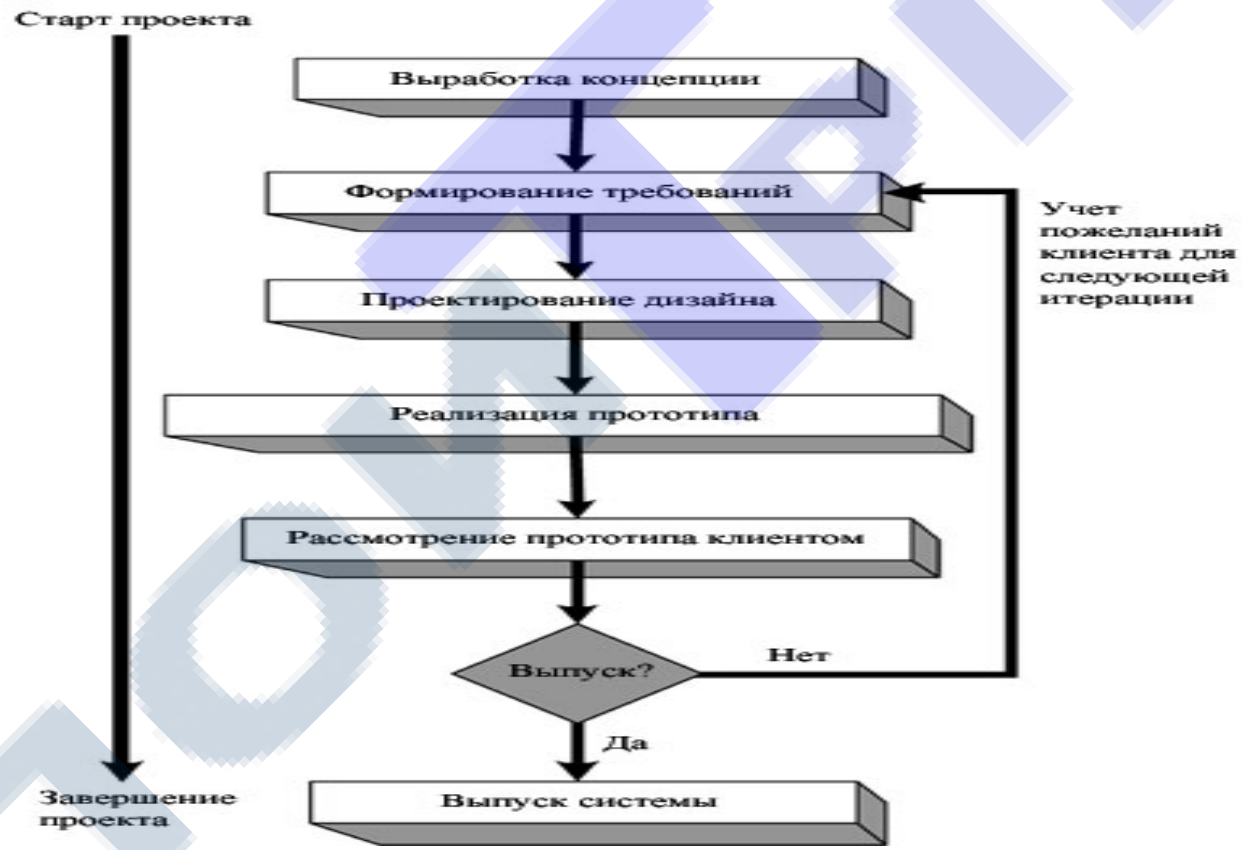
ИНКРЕМЕНТНАЯ СТРАТЕГИЯ: ОБЛАСТЬ ПРИМЕНЕНИЯ

1. Проекты, в которых большинство требований можно сформулировать заранее, но часть из них могут быть уточнены через определенный период времени
2. Сложные проекты с заранее сформулированными требованиями
3. При необходимости быстро поставить на рынок продукт, имеющий базовые функциональные свойства
4. Проекты с низкой или средней степенью рисков.

ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ: ХАРАКТЕРИСТИКА

- представляет собой многократный проход этапов разработки
- основана на частичном определении требований к разрабатываемому программному средству в начале процесса разработки
- требования постепенно уточняются в последовательных циклах разработки
- результат каждого цикла разработки представляет собой очередную поставляемую версию программного средства.

ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ: ХАРАКТЕРИСТИКА



ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ: ДОСТОИНСТВА

1. Возможность уточнения и внесения новых требований в процессе разработки
2. Пригодность промежуточного продукта для использования
3. Возможность управления рисками
4. Обеспечение широкого участия пользователя в проекте, начиная с ранних этапов, что минимизирует возможность разногласий между заказчиками и разработчиками и обеспечивает создание продукта высокого качества.

ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ: НЕДОСТАТКИ

1) Неизвестность точного количества необходимых итераций и сложность определения критериев для продолжения процесса разработки на следующей итерации
2) Сложность планирования и управления проектом
3) Необходимость активного участия пользователей в проекте, что реально не всегда осуществимо
4) Необходимость в мощных инструментальных средствах и методах прототипирования
5) Возможность сдвига решения трудных проблем на последующие циклы, что может привести к несоответствию полученных продуктов требованиям заказчиков.

ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ: ОБЛАСТЬ ПРИМЕНЕНИЯ

1. Проекты, для которых требования слишком сложны, неизвестны заранее, непостоянны или требуют уточнения
2. Сложные проекты, в том числе:
   - большие долгосрочные проекты
   - проекты по созданию новых, не имеющих аналогов ПС или систем
   - проекты со средней и высокой степенью рисков
   - проекты, для которых нужна проверка концепции, демонстрация технической осуществимости или промежуточных продуктов
3. Проекты, использующие новые технологии.

МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ЛЕКЦИЯ № 3

Ходить по воде и разрабатывать программы согласно ТЗ очень просто, если они заморожены

ПЛАН ЛЕКЦИИ

1. Модели ЖЦ ПО, реализующоие каскадную стратегию
2. Модель быстрой разработки приложений
3. Инкрементная модель экстремального программирования
4. Спиральная модель

0. ОПРЕДЕЛЕНИЕ МОДЕЛИ ЖЦ ПО

Модель ЖЦ – структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, организуемых в стадии, выполняемых на протяжении ЖЦ.

Процесс – совокупность взаимосвязанных или взаимодействующих видов деятельности (работ), преобразующих входы в выходы.

Выход процесса – наблюдаемый результат успешного достижения цели процесса:

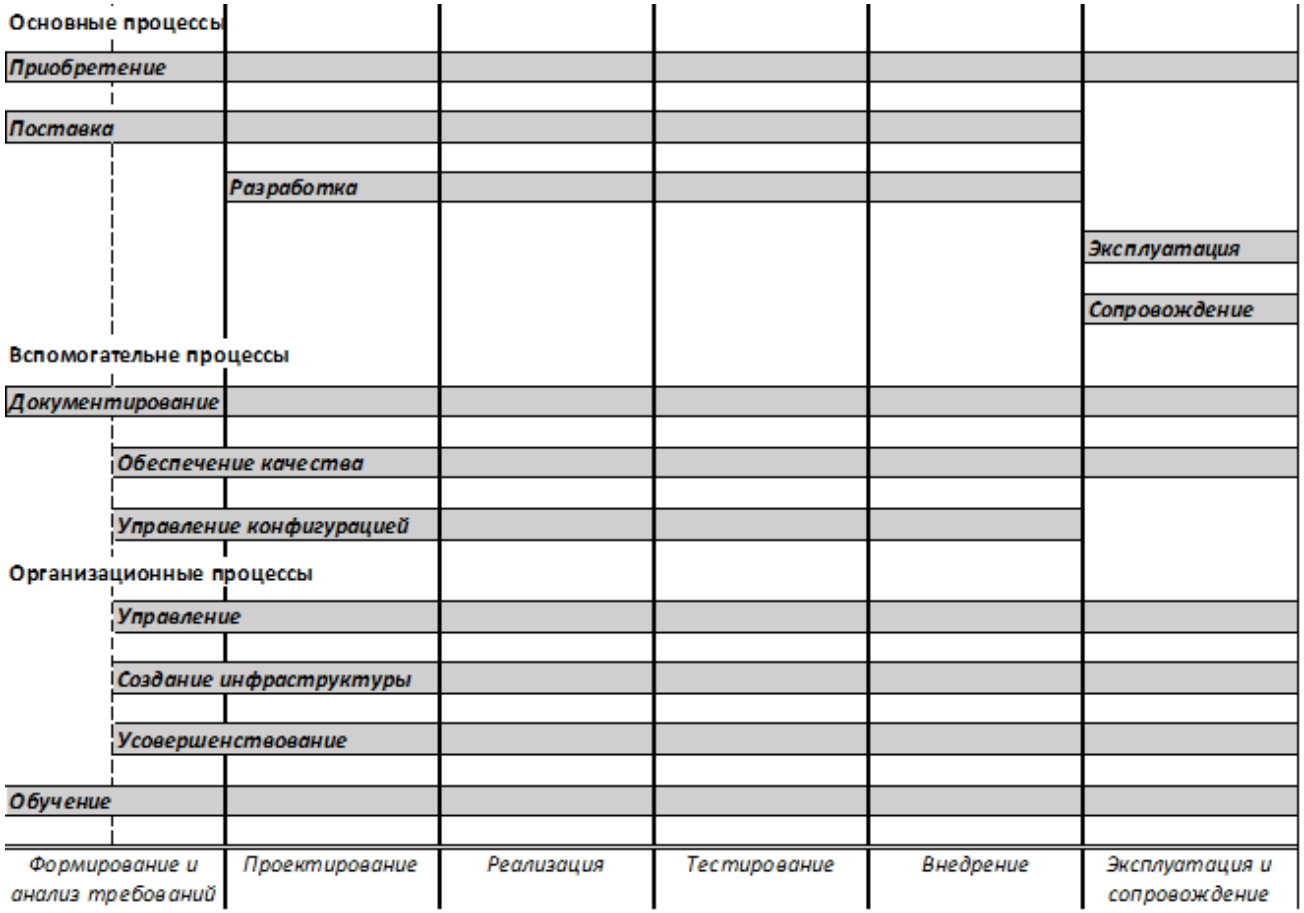
- изготовление какого-либо артефакта;
- существенное изменение состояния;
- удовлетворение заданных ограничений, например требований, конечных целей и т.п.

ISO/IEC/IEEE 12207:2017

«Системная и программная инженерия. Процессы жизненного цикла программных средств» (Systems and software engineering – Software life cycle processes)

- Стандарт не требует использования какой-либо конкретной модели жизненного цикла.
- Стандарт требует, чтобы в каждом проекте определялась подходящая модель жизненного цикла.
- Стандарт не содержит требований использования какой-либо заданной совокупности стадий.
- Последовательность процессов не предполагает какой-либо зависимости от времени.
- Пользователь настоящего стандарта может самостоятельно выбирать и назначать процессы, виды деятельности и задачи как наиболее подходящие и эффективные.

СТАДИИ И ВЫПОЛНЯЕМЫЕ НА НИХ ПРОЦЕССЫ

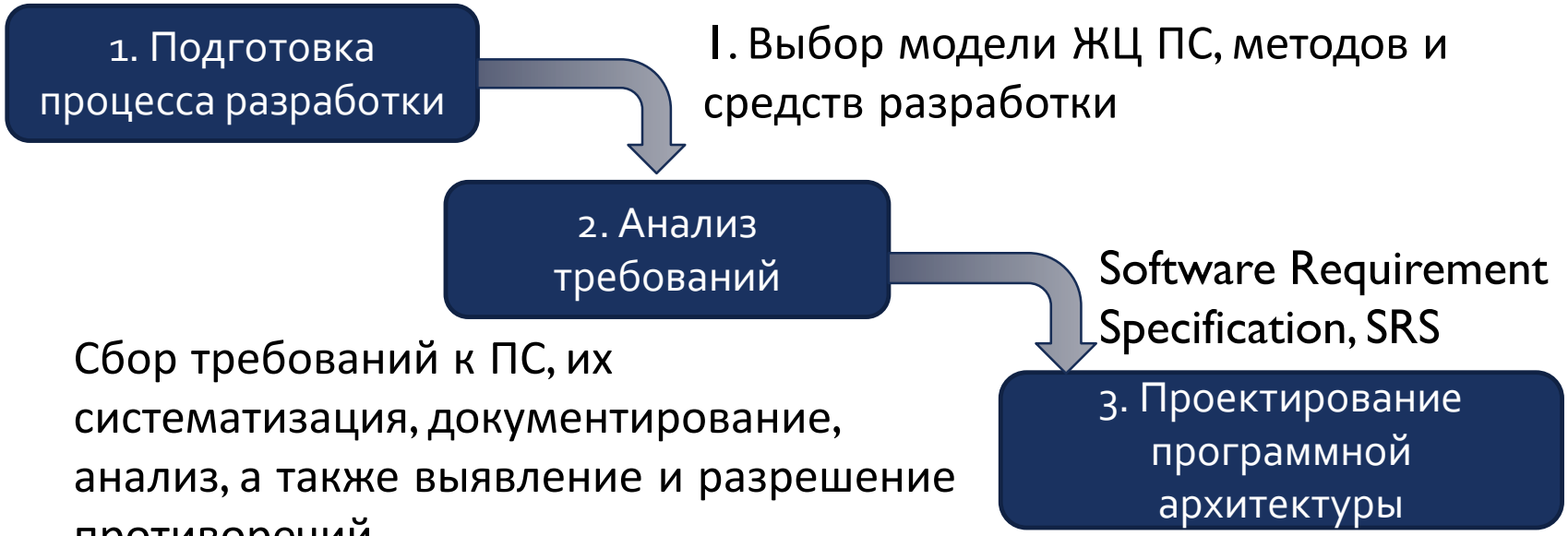


- Состав стадий и выполняемые на каждой стадии процессы зависят от модели ЖЦ ПО.

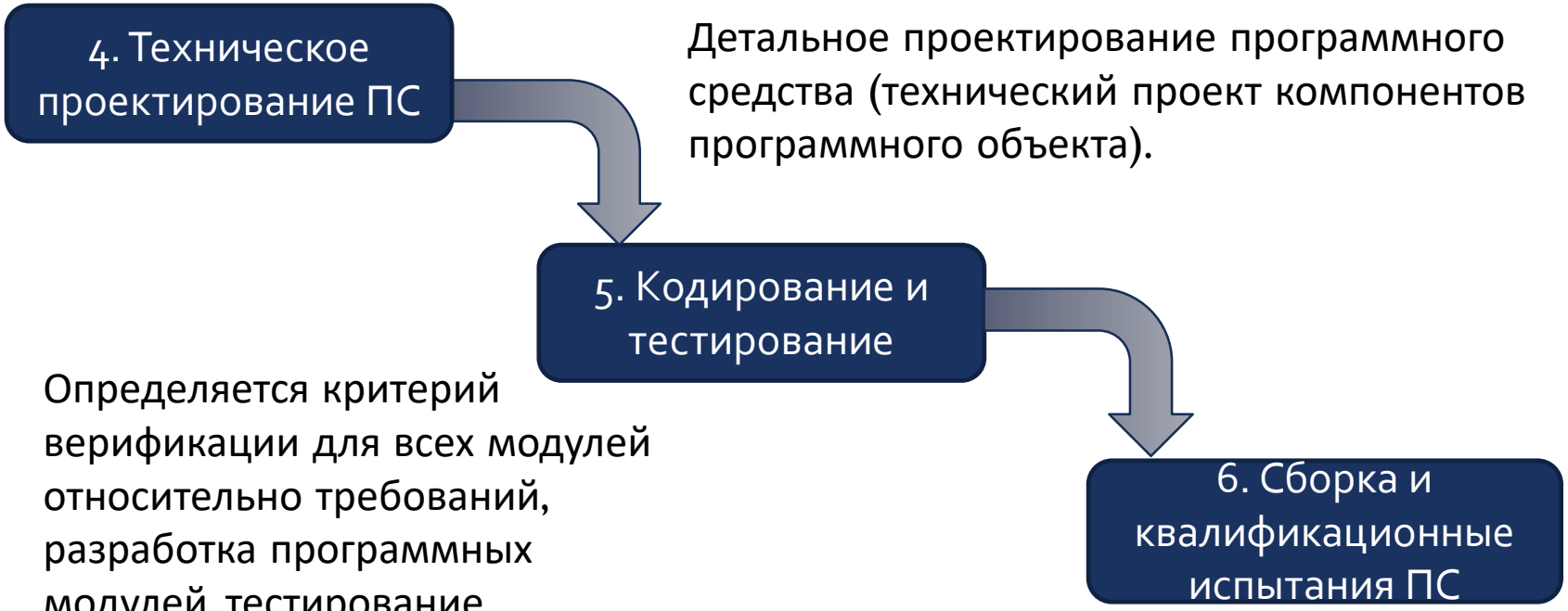
1. МОДЕЛИ ЖЦ ПО: КАСКАДНАЯ СТРАТЕГИЯ

- Классическая каскадная модель
- Каскадная модель с обратными связями
- V-образная модель

КЛАССИЧЕСКАЯ КАСКАДНАЯ МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА ПО



КЛАССИЧЕСКАЯ КАСКАДНАЯ МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА ПО



КАСКАДНАЯ МОДЕЛЬ С ОБРАТНЫМИ СВЯЗЯМИ

- Организация обратных связей между любыми шагами каскадной модели –> возможность исправления продуктов предыдущих шагов процесса разработки
- Сложности планирования и финансирования проекта, достаточно высокий риск нарушения графика разработки по сравнению с классической каскадной моделью

V-ОБРАЗНАЯ МОДЕЛЬ

- Связи между деятельностью по разработке планов испытаний и тестирования и деятельностью по подтверждению результатов соответствующих этапов.

2. МОДЕЛЬ БЫСТРОЙ РАЗРАБОТКИ ПРИЛОЖЕНИЙ : ХАРАКТЕРИСТИКА

RAD (Rapid Application Development)

Под RAD-разработкой обычно понимается процесс разработки, содержащий 3 элемента:

- небольшую команду разработчиков (до 10 человек);
- короткий, но тщательно проработанный производственный график (от 2 до 6 месяцев);
- повторяющийся цикл

2. МОДЕЛИ БЫСТРОЙ РАЗРАБОТКИ ПРИЛОЖЕНИЙ : ХАРАКТЕРИСТИКА

RAD (Rapid Application Development)

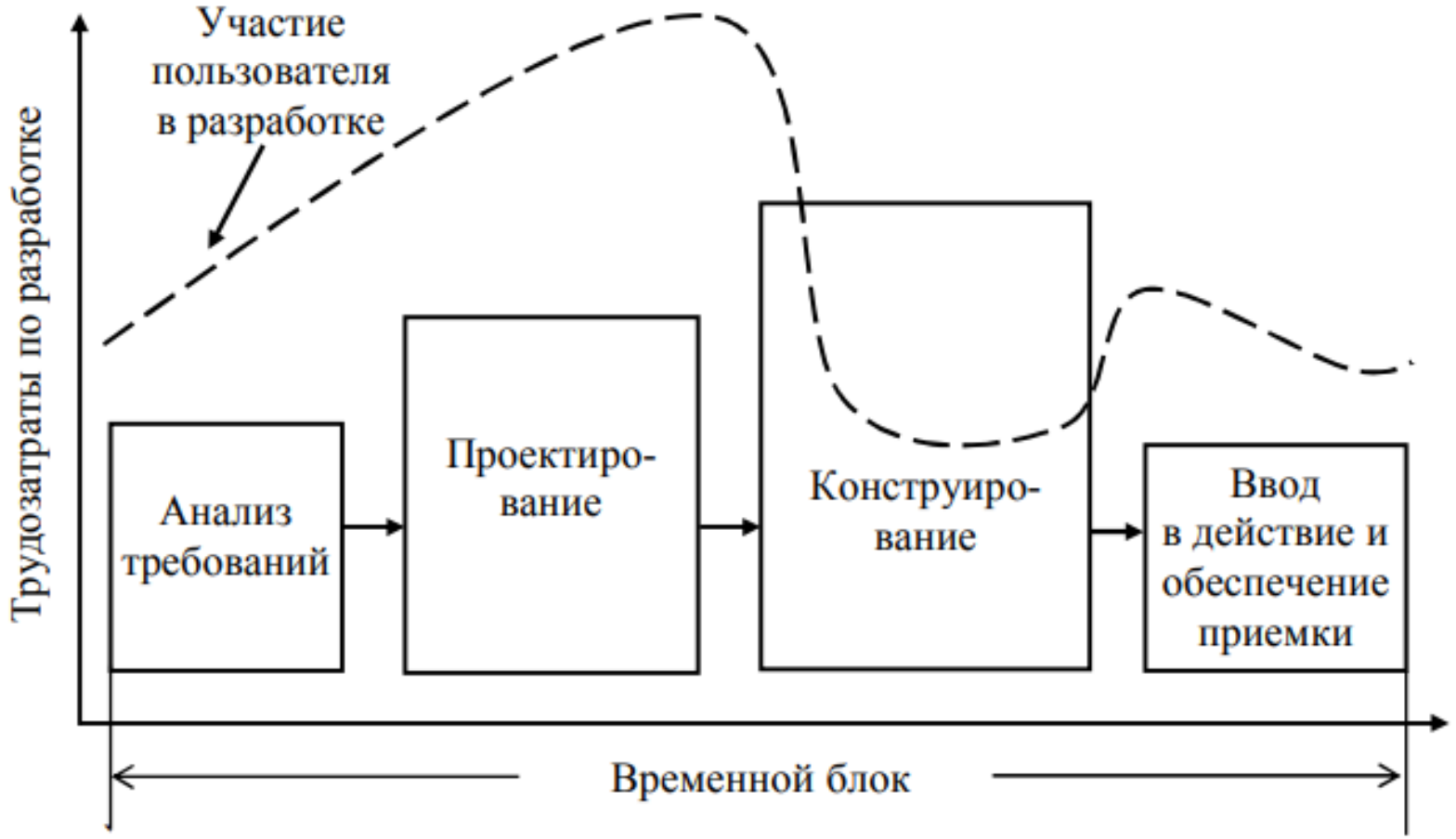
1. Эволюционная или инкрементная стратегии разработки

2. Мощные инструментальные средства разработки – визуальные среды проектирования и программирования

3. Прототипирование

Под **прототипом** понимается действующий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого ПО

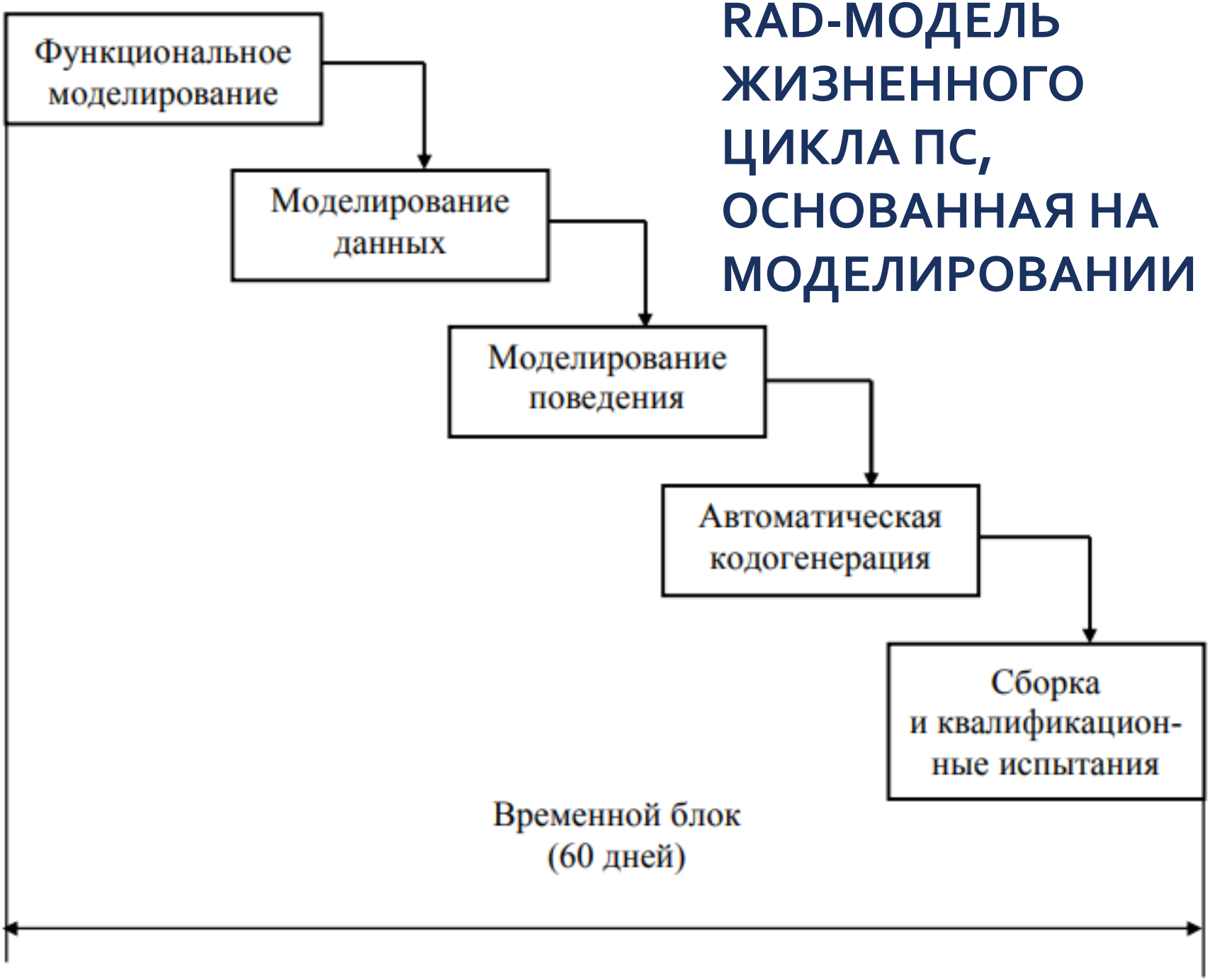
БАЗОВАЯ RAD-МОДЕЛЬ



RAD-МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА ПС, ОСНОВАННАЯ НА МОДЕЛИРОВАНИИ

1) Функциональное или бизнес-моделирование. Моделируется информационный поток между бизнес-функциями;
2) Моделирование данных или информационное моделирование. Информационный поток –> набор объектов данных (свойства, атрибуты) –> отношения между объектами;
3) Моделирование обработки (поведения). Определяются преобразования объектов данных, обеспечивающие реализацию бизнес функций;
4) Автоматическая кодогенерация;
5) Тестирование и объединение.

RAD-МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА ПС, ОСНОВАННАЯ НА МОДЕЛИРОВАНИИ



RAD-МОДЕЛЬ: ДОСТОИНСТВА

1) сокращение продолжительности цикла разработки и всего проекта в целом, сокращение количества разработчиков, а следовательно, и стоимости проекта за счет использования мощных инструментальных средств;
2) сокращение риска, связанного с выполнением графика, за счет использования принципа временного блока и связанное с этим упрощение планирования;
3) сокращение риска, связанного с неудовлетворенностью заказчика разработанным продуктом, за счет его привлечения на постоянной основе к циклу разработки;
4) возможность повторного использования существующих компонентов.

RAD-МОДЕЛЬ: НЕДОСТАТКИ

1) необходимость в высококвалифицированных разработчиках, умеющих работать с инструментальными средствами разработки;
2) возможность применения только для систем или программных средств, для которых отсутствует требование высокой производительности;
3) жесткость временных ограничений на разработку прототипа;
4) сложность ограничения затрат и определения сроков завершения работы над проектом
5) неприменимость в условиях высоких технических рисков, при использовании новых технологий.

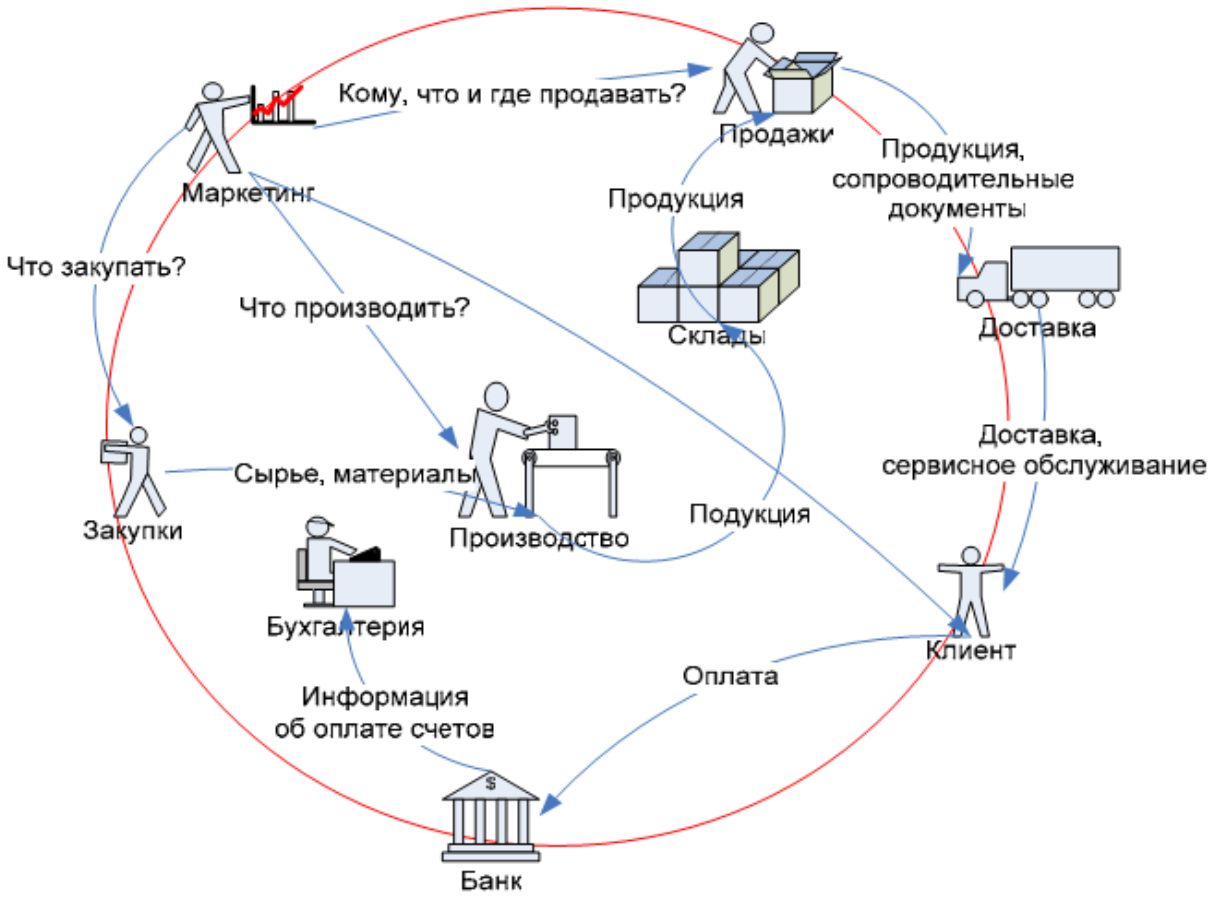
RAD-МОДЕЛЬ: ОБЛАСТЬ ПРИМЕНЕНИЯ

1) при разработке систем и продуктов, для которых характерно хотя бы одно из следующих свойств:
   - поддаются моделированию
   - имеют небольшой размер
   - имеют низкую производительность
   - относятся к известной разработчикам предметной области
   - являются информационными системами
   - требования для них хорошо известны
   - имеются компоненты пригодные к повторному использованию;

RAD-МОДЕЛЬ: ОБЛАСТЬ ПРИМЕНЕНИЯ

2) если пользователь может принимать постоянное участие в процессе разработки;
3) если в проекте заняты разработчики, обладающие достаточными навыками в использовании инструментальных средств разработки;
4) при выполнении проектов в сокращенные сроки (как правило, не более чем за 60 дней);
5) при разработке ПС, для которых требуется быстрое наращивание функциональных возможностей на последовательной основе;
6) при невысокой степени технических рисков;
7) в составе других моделей жизненного цикла.

RAD-МОДЕЛЬ: ОБЛАСТЬ ПРИМЕНЕНИЯ



- Функциональное моделирование

RAD-МОДЕЛЬ: ОБЛАСТЬ ПРИМЕНЕНИЯ

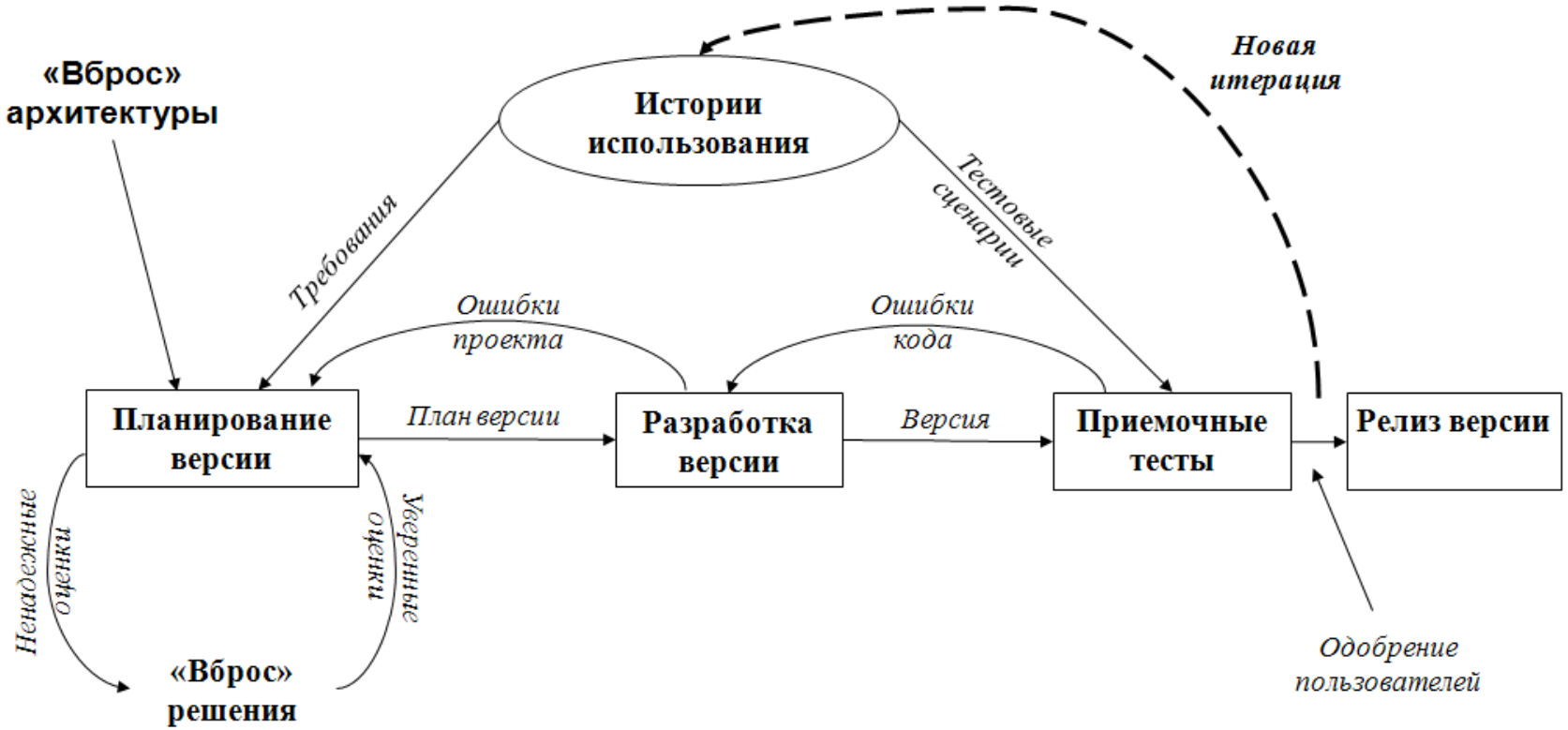
- Декомпозиция

RAD-МОДЕЛЬ: ОБЛАСТЬ ПРИМЕНЕНИЯ

3. ИНКРЕМЕНТНАЯ МОДЕЛЬ ЭКСТРЕМАЛЬНОГО ПРОГРАММИРОВАНИЯ

- Современная реализация инкрементной стратегии — экстремальное программирование XP (Кент Бек, 1999)
- Основными целями XP ***повышение*** качества программного продукта и ***резкое сокращение сроков разработки продукта***.

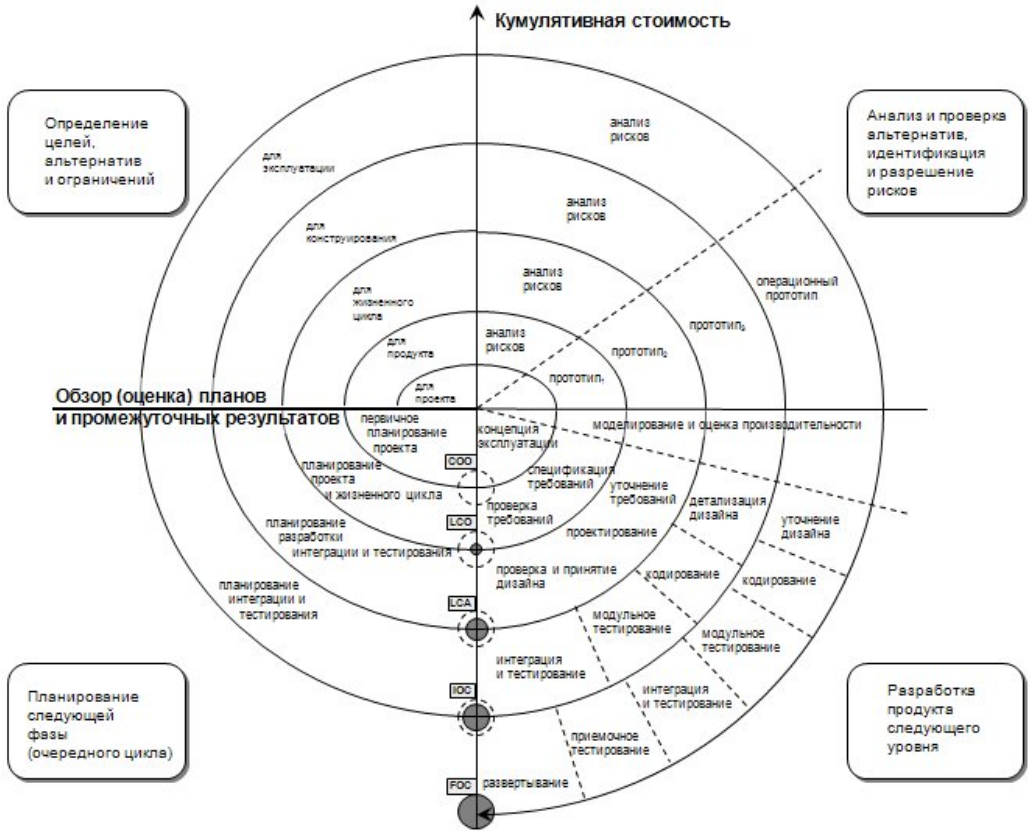
ИНКРЕМЕНТНАЯ МОДЕЛЬ ЭКСТРЕМАЛЬНОГО ПРОГРАММИРОВАНИЯ



ИНКРЕМЕНТНАЯ МОДЕЛЬ ЭКСТРЕМАЛЬНОГО ПРОГРАММИРОВАНИЯ

- **Итеративность.** Разработка ведется короткими итерациями при наличии активной взаимосвязи с заказчиком.
- **Простота решений.** Принимается первое простейшее рабочее решение.
- **Интенсивная разработка малыми группами** (не больше 10 человек) **и парное программирование** (когда два программиста вместе создают код на одном общем рабочем месте).
- **Обратная связь с заказчиком**, представитель которого фактически вовлечен в процесс разработки.
- **Достаточная степень смелости** и желание идти на риск.

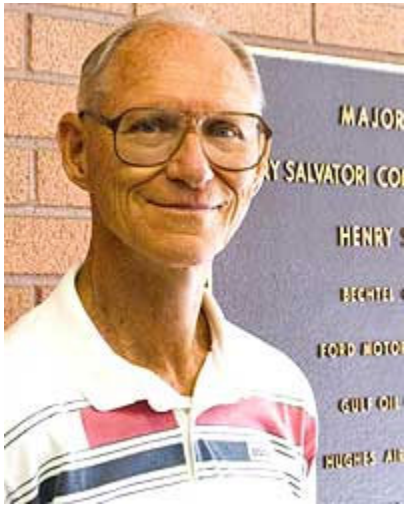
СПИРАЛЬНАЯ МОДЕЛЬ (Б.БОЭМ, 1988) ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ



Оригинальная спиральная модель жизненного цикла разработки по Боэму

- Отличительная особенность - специальное внимание ***рискам***, влияющим на организацию жизненного цикла.

СПИРАЛЬНАЯ МОДЕЛЬ (Б.БОЭМ, 1988) ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ



Барри Боэм

Главное достижение спиральной модели состоит в том, что она предлагает спектр возможностей адаптации удачных аспектов существующих моделей процессов жизненного цикла. В то же время, ориентированный на риски подход позволяет избежать многих сложностей, присутствующих в этих моделях. В определенных ситуациях спиральная модель становится эквивалентной одной из существующих моделей. В других случаях она обеспечивает возможность наилучшего соединения существующих подходов в контексте данного проекта.

СПИРАЛЬНАЯ МОДЕЛЬ (Б.БОЭМ, 1988) ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ

- Дефицит специалистов.
- Нереалистичные сроки и бюджет.
- Реализация несоответствующей функциональности.
- Разработка неправильного пользовательского интерфейса.
- “Золотая сервировка”, 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 знаний.

СПИРАЛЬНАЯ МОДЕЛЬ (Б.БОЭМ, 1988) ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ

- **Квадрант I** – анализ целей, альтернативных вариантов и ограничений – определяются рабочие характеристики, выполняемые функции, стабильность (возможность внесения изменений), аппаратно-программный интерфейс продукта разработки данной фазы или цикла.
- **Квадрант II** – оценка альтернативных вариантов, идентификация и разрешение рисков. Выполняется прототипирование как основа для работ следующего квадранта.
- **Квадрант III** – разработка продукта текущего уровня (разработка и тестирование исходных текстов программ, сборка, тестирование и квалификационные испытания продукта или системы и т.п.)
- **Квадрант IV** – планирование следующей фазы – решение о переходе на цикл следующей фазы разработки или выполнении еще одного цикла текущей фазы разработки

СПИРАЛЬНАЯ МОДЕЛЬ (Б.БОЭМ, 1988) ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ

- **A. Фаза разработки концепции** (соответствует первому витку спирали)
- **B**. **Фаза анализа требований** (соответствует второму витку спирали)
- **C**. **Фаза проектирования** системы/программного продукта (соответствует третьему витку спирали)
- **D. Фаза реализации** (технического проектирования, программирования и сборки)
- **E. Фаза сопровождения и расширения** функциональных возможностей (соответствует пятому витку спирали).

СПИРАЛЬНАЯ МОДЕЛЬ (Б.БОЭМ, 1988) ЭВОЛЮЦИОННАЯ СТРАТЕГИЯ

Движение по спирали:

- каждый виток спирали (итерация) – совокупность стадий каскадной модели
- каждый виток спирали соответствует созданию фрагмента или версии ПО
- с каждой итерацией по спирали строятся все более полные версии ПС.

СПИРАЛЬНАЯ МОДЕЛЬ (Б.БОЭМ, 1988) ПРЕИМУЩЕСТВА

1) Упрощение изменений в проекта при изменении требований заказчика.
2) Уменьшение уровня рисков.
3) Гибкость в управлении проектом.
4) Итерационный подход упрощает повторное использование компонентов.
5) Спиральная модель позволяет получить более надежную и устойчивую систему, т.к. ошибки и слабые места обнаруживаются и исправляются на каждой итерации

СПИРАЛЬНАЯ МОДЕЛЬ (Б.БОЭМ, 1988) НЕДОСТАТКИ

1) Сложность анализа и оценки рисков при выборе вариантов –> требует искусного управления
2) Сложность поддержания версий продукта (хранение версий, возврат к ранним версиям, комбинация версий) –> необходима поддержка целостности документации;
3) Сложность в определении момента перехода на следующий этап –> временные ограничения на каждый из этапов жизненного цикла.

СПИРАЛЬНАЯ МОДЕЛЬ (Б.БОЭМ, 1988) ОБЛАСТЬ ПРИМЕНЕНИЯ

- Когда пользователи не уверены в своих потребностях или когда требования слишком сложны и могут меняться в процессе выполнения проекта и необходимо прототипирование для анализа и оценки требований
- Когда достижение успеха не гарантировано и необходима оценка рисков продолжения проекта
- Когда проект является сложным, дорогостоящим и обоснование его финансирования возможно только в процессе его выполнения
- Когда речь идет о применении новых технологий, что связано с риском их освоения и достижения ожидаемого результата
- При выполнении очень больших проектов, которые в силу ограниченности ресурсов можно делать только по частям.

УПРОЩЕННЫЕ ВАРИАНТЫ СПИРАЛЬНОЙ МОДЕЛИ

- Модель Института качества SQI
- Модель Института Управления проектами PMI
- Модель «win-win»
- Спиральная модель Консорциума по вопросам разработки программного обеспечения

ВЫБОР МОДЕЛИ ЖЦ ПО

Критерии категории требований	Каскадная	V-модель	Инкрементная	Эволюционная
Являются ли требования к проекту легко определимыми и реализуемыми?	Да	Да	Нет	Нет
Могут ли требования быть сформулированы в начале ЖЦ?	Да	Да	Да	Нет
Часто ли будут изменяться требования на протяжении ЖЦ?	Нет	Нет	Нет	Да
Нужно ли реализовать основные требования на ранних этапах разработки?	Нет	Нет	Да	Да

ВЫБОР МОДЕЛИ ЖЦ ПО

Критерии категории команды разработчиков проекта	Каскадная	V-модель	Инкрементная	Эволюционная
Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Нет	Нет	Нет	Да
Изменяются ли роли участников проекта на протяжении ЖЦ?	Нет	Нет	Да	Да
Является ли структура процесса разработки более значимой для разработчиков, чем гибкость?	Да	Да	Да	Нет