Т.П. Стасеня
О.Г. Мандрик

ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Курс лекций

Витебск
2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«Витебский государственный технологический университет»

Т.П. Стасеня

О.Г. Мандрик

Технологии проектирования программного обеспечения информационных систем

Курс лекций
для слушателей специальности переподготовки
1-40 01 73 «Программное обеспечение информационных систем»

Витебск
2017

УДК 004.41/42
ББК 32.973-018
С 76

Рецензент:
ведущий программист ЦИТ УО «ВГТУ» Любочко Н.С.

Рекомендовано к изданию редакционно-издательским советом УО «ВГТУ», протокол № 8 от 26.10.2016.

Стасеня, Т. П.

С 76 Технологии проектирования программного обеспечения информационных систем : курс лекций / Т. П. Стасеня, О. Г. Мандрик. – Витебск : УО «ВГТУ», 2017. – 45 с.

ISBN 978-985-481-457-5

В материалах курса лекций рассмотрены основные приемы по разработке информационных систем, которые включают в себя работы по созданию информационного программного обеспечения и его компонентов в соответствии с заданными требованиями. Лекционный материал предназначен для использования при выполнении лабораторных работ и подготовке к экзамену по предмету «Технологии проектирования программного обеспечения информационных систем» слушателями специальности 1-40 01 73 «Программное обеспечение информационных систем» и будет полезен для студентов других специальностей при изучении курсов «Информатика», «Информационные технологии».

УДК 004.41/42
ББК 32.973-018

ISBN 978-985-481-457-5

Оглавление

Тема 1. Введение. Основные понятия и определения

Разработка информационной системы включает в себя все работы по созданию информационного программного обеспечения и его компонентов в соответствии с заданными требованиями. Разработка также включает оформление проектной и эксплуатационной документации; подготовку материалов, необходимых для тестирования разработанных программных продуктов; разработку материалов, необходимых для обучения персонала.

Технологии разработки и проектирования ПО – это:

- совокупность процессов и методов создания программного продукта;
- система инженерных принципов для создания экономичного ПО, которое надежно и эффективно работает в реальных компьютерах;
- система инженерных принципов для создания экономичного ПО с заданными хартеристиками качества.

Любая технология разработки ПО базируется на некоторой методологии.

Методология – это система принципов и способов организации процессов разработки программ.

Цель методологии разработки ПО – внедрение методов разработки программ, обеспечение достижения соответствия характеристик качества.

В настоящее время широкую известность приобрели два базовых принципа разработки программных средств (ПС): ***модульный принцип*** **и** ***объектно-ориентированный принцип.***

Система – комплекс, состоящий из процессов, программных средств, устройств и персонала, обладающий возможностью удовлетворять установленным потребностям или целям.

Под ***жизненным циклом*** (ЖЦ) программного средства или системы подразумевается совокупность процессов, работ и задач, включающая в себя разработку, эксплуатацию и сопровождение ПС или системы, охватывающая их жизнь от формулирования концепции до прекращения использования [1].

Разработка является одним из важнейших процессов жизненного цикла информационной системы и, как правило, включает в себя стратегическое планирование, анализ, проектирование и реализацию (программирование).

Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами.

Классификация процессов ЖЦ ПС

Структура ЖЦ ПС

ЖЦ состоит из процессов, процессы из работ, работы из задач.

Процессы ЖЦ ПС делятся на три группы: основные; вспомогательные;

организационные.

Основные процессы: заказ; поставка; разработка; эксплуатация; сопровождение.

Процесс разработки содержит ***тринадцать работ***:

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) квалификационные испытания программных средств;
10) сборка системы;
11) квалификационные испытания системы;
12) ввод в действие программных средств;
13) обеспечение приемки программных средств.

В процессе **разработки** существуют **2 вида работ: системные и программные.**

К системным работам относятся 2, 3, 10, 11 работы:

2) анализ требований к системе;
3) проектирование системной архитектуры;
10) сборка системы;
11) квалификационные испытания системы.

Программные работы – 4 – 9 работы:

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) квалификационные испытания программных средств.

Организационные процессы: управление; создание инфраструктуры; усовершенствование; обучение.

Вспомогательные процессы: документирование; управление конфигурацией; обеспечение качества; верификация; аттестация; совместный анализ; аудит; решение проблем.

Модель жизненного цикла – это совокупность процессов, работ и задач ЖЦ, отражающая их взаимосвязь и последовательность выполнения [1].

Модель жизненного цикла информационной системы представляет собой некоторую структуру, определяющую последовательность

осуществления процессов, действий и задач, выполняемых на протяжении жизненного цикла информационной системы, а также взаимосвязи между этими процессами, действиями и задачами.

Модель жизненного цикла зависит от специфики информационной системы и условий, в которых она создается и функционирует. Выбор конкретной модели жизненного цикла и метода разработки информационных систем зависит от многих факторов и привязки к определенной предметной области.

Тема 2. Базовые стратегии и модели разработки ПС и систем

2.1 Базовые стратегии

Первая модель разработки программного продукта. Основана на принципе – «кодирование – устранение недостатков».

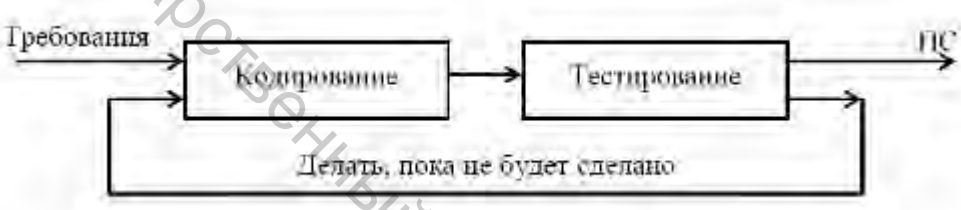

Рисунок 1 – Первая модель разработки ПО

Недостатки модели: неструктурированность процесса разработки программных средств; ориентация на индивидуальные знания и умения программиста; сложность управления и планирования проекта; большая длительность и стоимость разработки; низкое качество программных продуктов; высокий уровень рисков проекта.

В настоящее время используется 3 базовых стратегии**:**

- **каскадная,**
- **инкрементная (пошаговая, циклическая),**
- **эволюционная**.

Выбор той или иной стратегии определяется характеристиками: проекта; требований к продукту; команды разработчиков; команды пользователей.

Каскадная стратегия **–** представляет собой однократный проход этапов разработки. Реализует метод нисходящего проектирования ПС и систем. Основана на полном определении всех требований к разрабатываемому продукту в начале процесса разработки. Возврат к выполненным этапам разработки не предусмотрен. Промежуточные результаты в качестве версий продукта не рассматриваются.

Модели, которые ее реализуют: каскадная и V-образная.

Достоинства:

1) стабильность требований в течение всего жизненного цикла разработки;

2) простота применения стратегии; необходимость только одного прохода этапов разработки;

3) простота планирования, контроля и управления проектом;

4) возможность достижения высоких требований к качеству проекта в случае отсутствия жестких ограничений затрат и графика работ;

5) доступность для понимания заказчиками.

Недостатки:

1) сложность четкого формулирования требований в начале жизненного цикла ПС и невозможность их динамического изменения на протяжении ЖЦ ПС;

2) последовательность линейной структуры процесса разработки; в результате возврат к предыдущим шагам для решения возникающих проблем приводит к увеличению затрат и нарушению графика работ;

3) проблемность финансирования проекта, связанная со сложностью единовременного распределения больших денежных средств;

4) непригодность промежуточного продукта для использования;

5) недостаточное участие пользователя в разработке системы или ПС; участие заказчика только в самом начале (при разработке требований) и в конце (во время приемочных испытаний), что приводит к невозможности предварительной оценки пользователем качества ПС или системы.

Области применения:

1) при разработке проектов с четкими, неизменяемыми в течение ЖЦ требованиями, понятными реализацией и техническими методиками;

2) при разработке проекта, ориентированного на построение системы или продукта такого же типа, которые уже разрабатывались разработчиками ранее;

3) при разработке проекта, связанного с созданием и выпуском новой версии уже существующего продукта или системы;

4) при разработке проекта, связанного с переносом уже существующего продукта на новую платформу;

5) при выполнении больших проектов, в которых задействовано несколько больших команд разработчиков.

Инкрементная стратегия разработки программных средств и систем представляет собой многократный проход этапов разработки с запланированными улучшениями результата. Данная стратегия основана на полном определении всех требований к разрабываемому продукту в начале процесса разработки. Однако полный набор требований реализуется постепенно в соответствии с планом.

Результат каждого цикла называется **инкрементом**. Первый инкремент реализует базов функции, последние – следующие функции. Функции на

каждом инкременте постепенно расширяются, пока не будет реализован весь набор. Различия между соседними инкрементами постепенно уменьшаются. Результат каждого цикла разработки можно рассматривать как поставление версии программного продукта. Особенность – большое количество циклов разработки при небольшом различии между соседними инкрементами и незначительное продолжение цикла. Икрементная стратегия базируется на использовании объединения каскадной стратегии и прототипирования.

Прототип – легко поддающаяся модификация и расширению рабочая модель предполагаемой системы или ПС, позволяющая пользователю получить представление о ключевых свойствах ПС или программы до их полной реализации.

Достоинства:

1) возможность получения функционального продукта после реализации каждого инкремента;

2) короткая продолжительность создания инкремента; это приводит к ускорению начального графика поставки и снижает затраты поставки ПП;

3) предотвращение реализации громоздких перечней требований; возможность учета изменившихся требований;

4) снижение риска неудачи и изменения требований по сравнению с каскадной моделью;

5) включение в процесс пользователей, что позволяет оценить самые важные функциональные возможности продукта на более ранних этапах разработки и приводит к повышению качества ПП, снижению затрат и времени на его разработку;

6) стабильность требований во время создания определенного инкремента, возможность учета изменяющихся требований.

Недостатки:

1) необходимость полного функционального определения системы или программного средства в начале жизненного цикла для обеспечения определения инкрементов, планирования и управления проектом;

2) возможность текущего изменения требований к системе или программному средству, которые уже реализованы в предыдущих инкрементах;

3) необходимость хорошего планирования и проектирования, грамотного распределения работы;

4) наличие тенденции к оттягиванию решения трудных проблем на поздние инкременты, что может нарушить график работ.

Области применения:

1) при разработке проектов, в которых большинство требований можно сформулировать заранее, но часть из них может быть сформулирована через определенный период времени;

2) при необходимости быстро поставить на рынок продукт, имеющий функциональные базовые свойства;

3) для выполнения проектов с большим периодом разработки (один год и более);

4) при разработке проектов с низкой или средней степенью риска;

5) при выполнении проекта с применением новой технологии.

Эволюционная стратегия разработки ПС и систем представляет собой многократный проход этапов разработки. Основан на частичном определении требований в начале процесса разработки. Требования постепенно уточняются в каждом последующем цикле. Результат – уточнение в очередной поставленной версии ПС. В общем случае для эволюционной стратегии характерно существенно меньшее количество циклов разработки при большой продолжительности цикла по сравнению с инкрементной моделью. Зачастую при реализации эволюционной стратегии используется прототипирование.

Достоинства:

1) возможность уточнения и внесения новых требований в процессе разработки;

2) пригодность для использования промежуточного продукта;

3) возможность обеспечения управления рисками;

4) реализация преимуществ инкрементной стратегии (выпуск версий, повторное использование результатов, сокращение графика работ);

5) обеспечение широкого участия пользователя в проекте, начиная с ранних этапов, что минимизирует возможность разногласий между заказчиками и разработчиками и обеспечивает создание продукта высокого качества;

6) уменьшение общих затрат на разработку за счет раннего решения проблем и меньшего объема доработок.

Недостатки:

1) сложность определения критериев для продолжения процесса разработки на следующей итерации, неизвестность точного количества необходимых итераций; это затрудняет планирование проекта и может вызвать задержку реализации конечной версии системы или программного средства;

2) сложность управления проектом; необходимость грамотного управления с целью ограничения количества итераций до разумного;

3) необходимость активного участия пользователей в проекте, что не всегда возможно в реальной жизни;

4) необходимость мощных инструментальных средств и методов прототипирования;

5) возможность отодвигания решения трудных проблем на последующие циклы, что может привести к несоответствию полученных продуктов требованиям заказчиков.

Области применения:

1) при разработке проектов, для которых требования слишком сложны, неизвестны заранее, непостоянны или их следует уточнить;

2) при разработке больших и долгосрочных проектов;
3) при разработке проектов используются новые технологии.

2.2 Модели ЖЦ разработки ПС

Каскадная модель

Классическая каскадная модель ЖЦ ПС и варианты ее реализации.

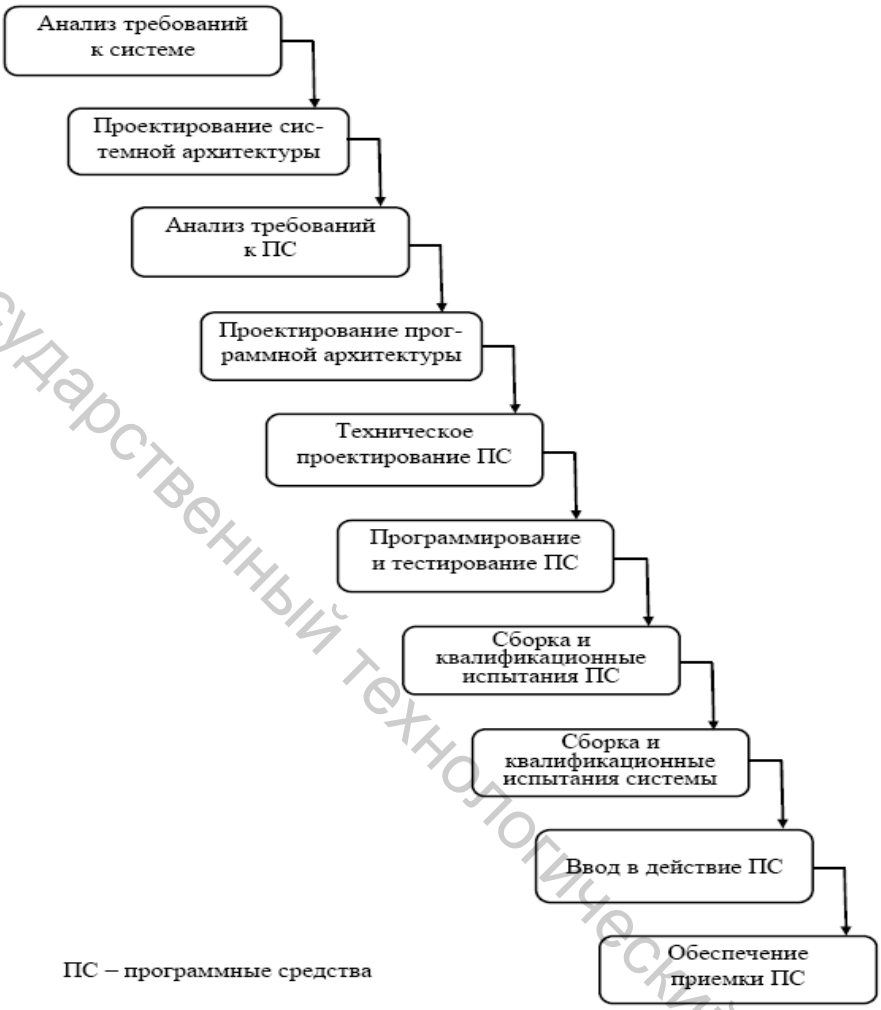

Рисунок 2 – Классическая каскадная модель процесса разработки по СТБ ИСО/МЭК 12207-2003

Каскадная модель обладает всеми достоинствами и недостатками, характерными для каскадной стратегии разработки. Ввиду сложности реализации на реальных проектах классической каскадной модели была создана каскадная модель с обратными связями.

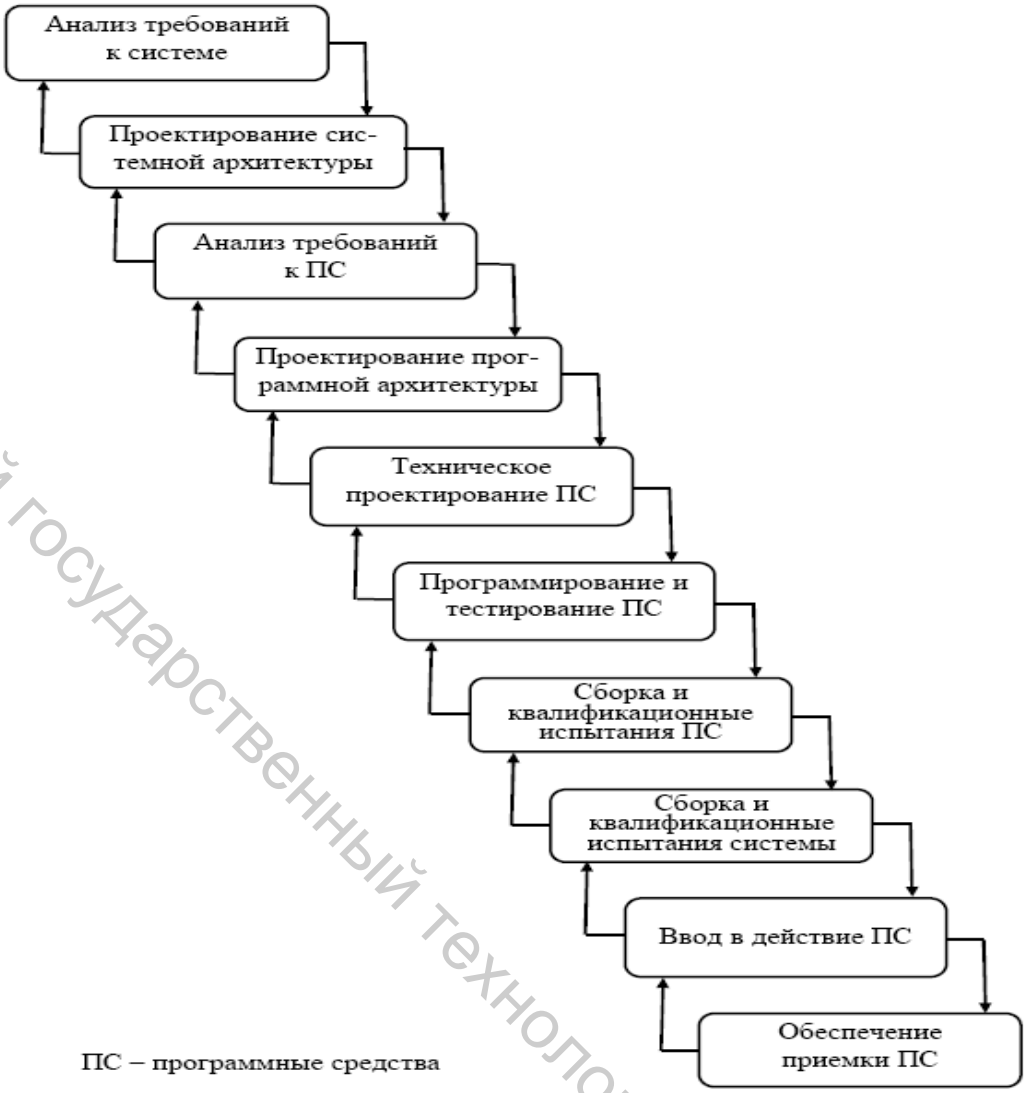

Рисунок 3 – Каскадная модель с обратными связями

V-образная модель жизненного цикла ПС.

Назначение – обеспечение планирования, тестирования системы и ПС на ранних стадиях проекта. Отличие от каскадной модели: здесь выделены связи между шагами предшествующими программированию и соответствующие видам тестирования и испытаний.

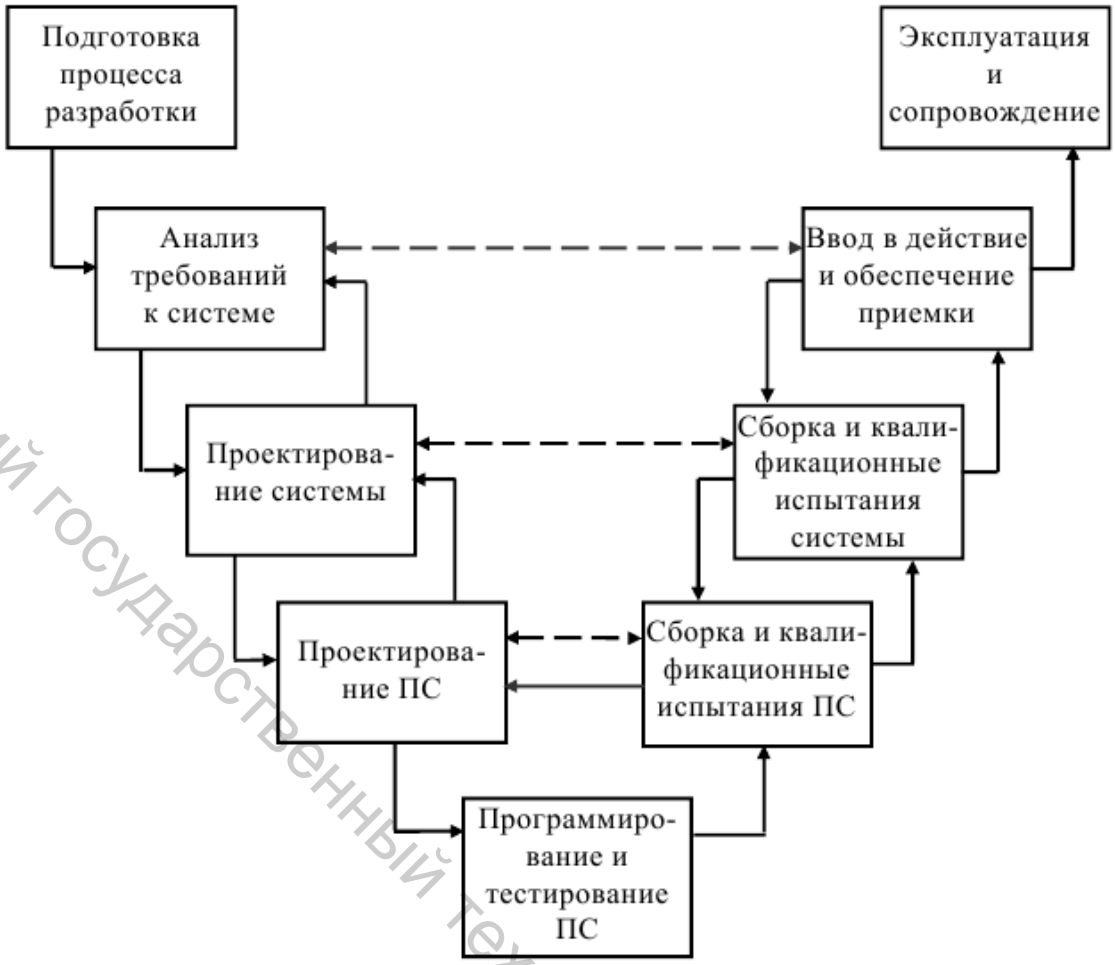

Рисунок 4 – V- образная модель разработки

На этапе 2 выполняются разработки плана ввода в действие и обеспечение приемки.

На 3 этапе помимо основных действий разрабатывается план квалификационных испытаний системы.

На 4 этапе разрабатывается план сборки ПС и план квалификационных испытаний ПС.

5 и 6 этапы – программирование.

На 7 этапе подтверждаются результаты этапа проектирования системы.

На 8 – выполняются приемочные испытания, их цели – проверка пользователем соответствия системы исходным требованиям к системе.

V-образная модель поддерживает каскадную стратегию разработки программных средств и систем. Поэтому она обладает всеми достоинствами данной стратегии.

Достоинства:

1) планирование на ранних стадиях разработки системы ее тестирования (испытаний);

2) упрощение аттестации и верификации всех промежуточных результатов разработки;

3) упрощение управления и контроля хода процесса разработки, возможность более реального использования графика проекта.

Недостатки:

1) поздние сроки тестирования требований;

2) отсутствие действий, направленных на анализ рисков.

Вариант реализации с обратными связями, т. е. возможность вернуться на предыдущий шаг. Все остальное как в каскадной модели.

Базовая RAD-модель быстрой разработки приложений жизненного цикла ПС

Модель быстрой разработки приложений – Rapid Application Development. Данная модель может поддерживать как инкрементную, так и эволюционную стратегию разработки системы или программного средства. Как правило, данная модель используется в составе другой модели для ускорения цикла разработки прототипа (версии) системы или ПС.

Основа RAD модели – использование *мощного инструмента средств разработки – CASE или языки 4 поколения.* Характерной чертой RAD-модели является короткое время перехода от анализа требований до создания полной системы или программного средства.

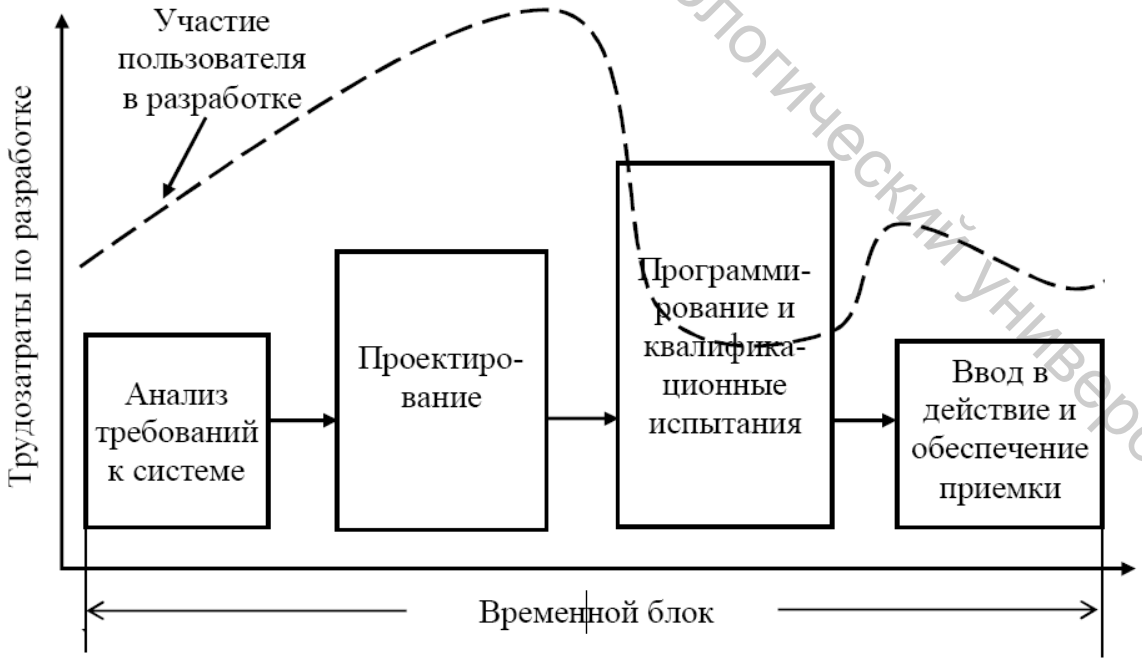

Рисунок 5 – Базовая **RAD-модель** для реализации отдельной итерации эволюционной модели

RAD-модель жизненного цикла ПС, основанная на моделировании.

.

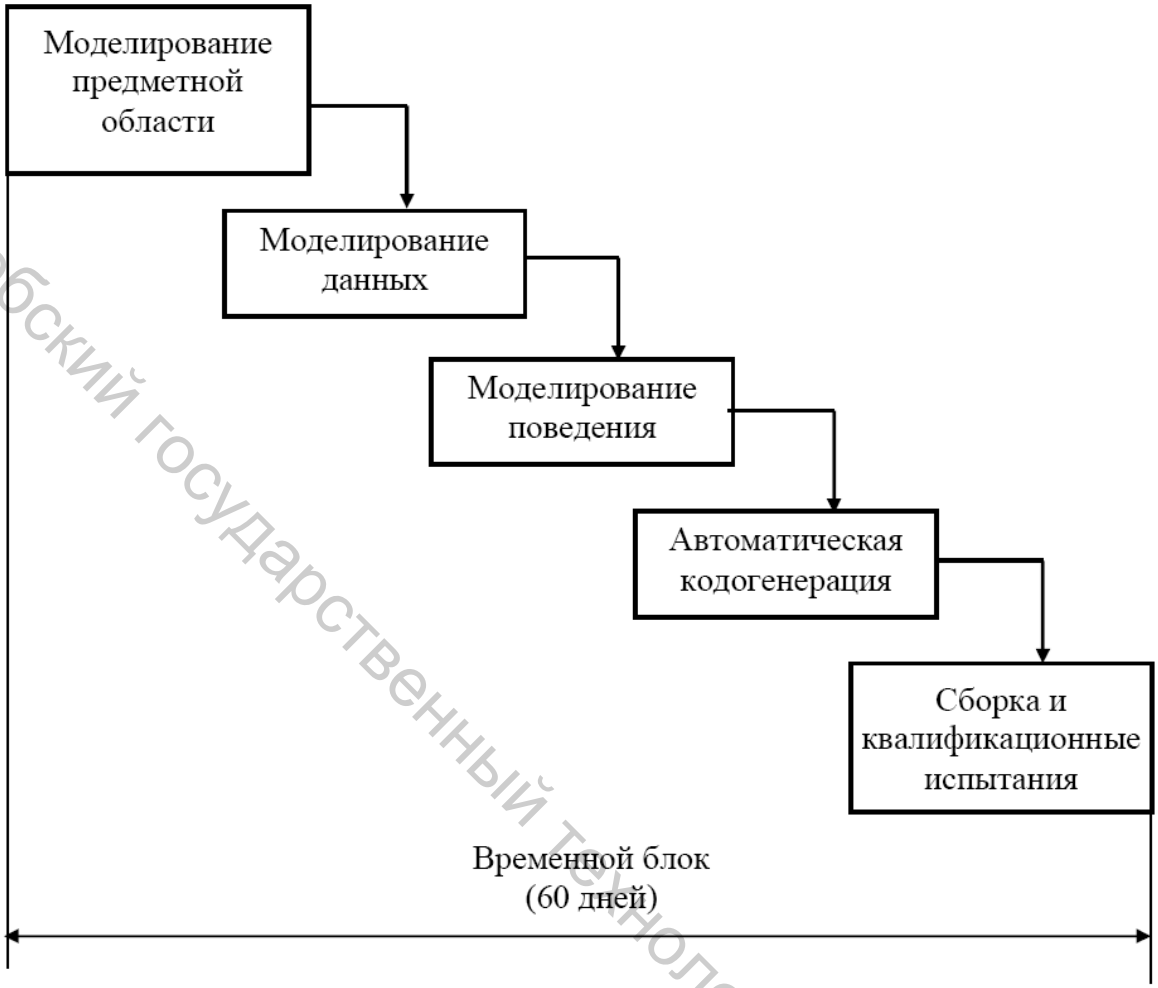

Рисунок 6 – **RAD-модель жизненного цикла ПС, основанная на моделировании**

Достоинства RAD-модели:

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 соблюдением 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; возрастание уверенности, что система будет соответствовать требованиям;

4) возможность повторного использования существующих компонентов.

RAD-модель быстрой разработки приложения по ГОСТ Р ИСО/МЭК ТО 15271-2002

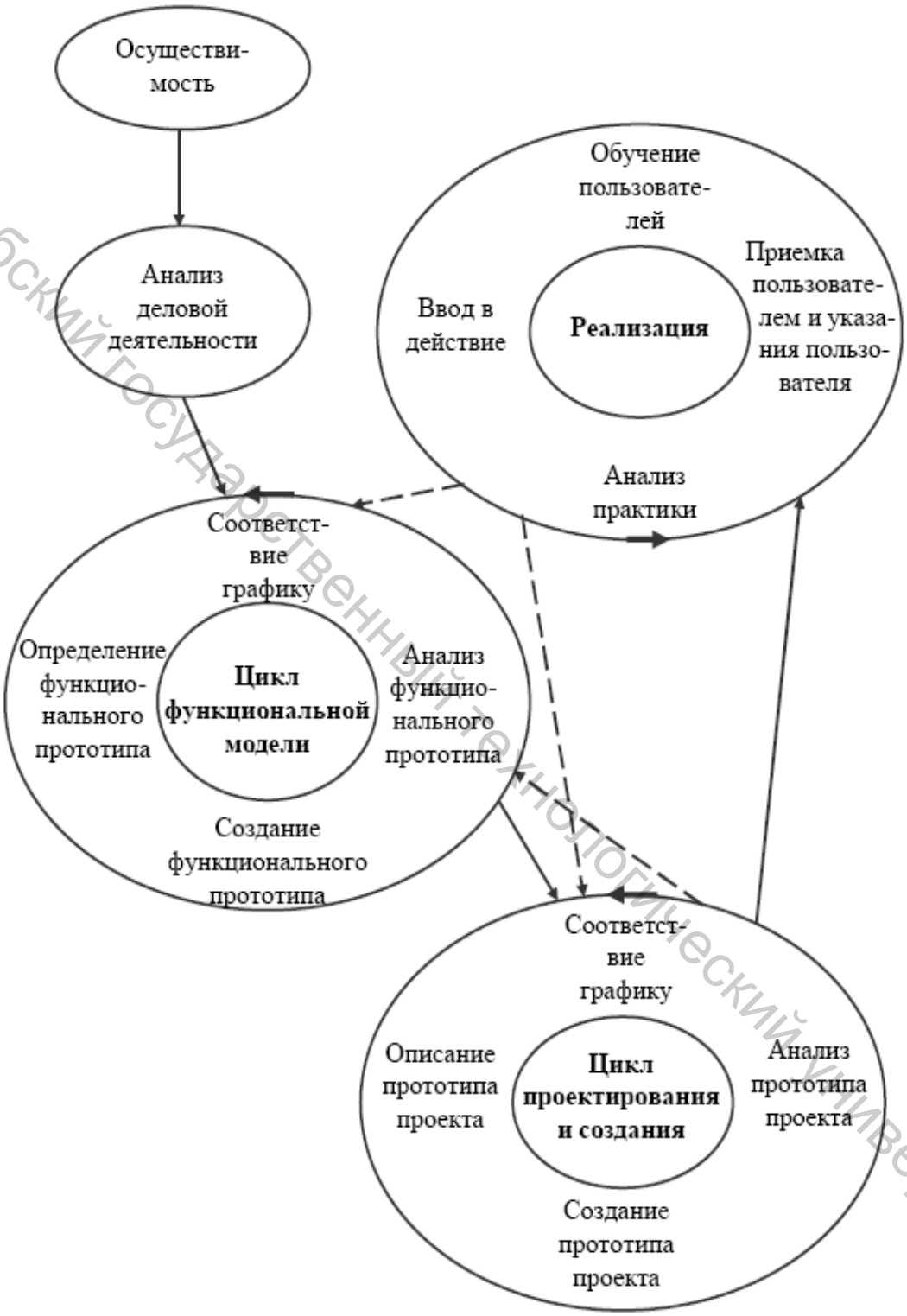

Рисунок 7 – RAD-модель быстрой разработки приложения по ГОСТ Р ИСО/МЭК ТО 15271-2002

Недостатки:

1) необходимость в постоянном участии пользователя в процессе разработки, что часто невыполнимо и в итоге сказывается на качестве конечного продукта;

2) необходимость в высококвалифицированных разработчиках, умеющих работать с инструментальными средствами разработки;

3) возможность применения только для систем или программных средств, для которых отсутствует требование высокой производительности;

4) жесткость временных ограничений на разработку прототипа;

5) сложность ограничения затрат и определения сроков завершения работы над проектом;

6) неприменимость в условиях высоких технических рисков, при использовании новых технологий.

Области применения:

1) при разработке систем и продуктов, которые удовлетворяют хотя бы одному из следующих свойств:

- поддаются моделированию;
- требования для них хорошо известны;
- предназначены для концептуальной проверки;
- являются некритическими;
- имеют небольшой размер;
- имеют низкую производительность;
- относятся к известной разработчикам предметной области;
- являются информационными системами;
- для них имеются пригодные к повторному использованию компоненты.

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 систем, для которых требуется быстрое наращивание функциональных возможностей на последовательной основе;

6) при разработке проектов, для которых затраты и соблюдение графика не являются очень важными;

7) при невысокой степени технических рисков.

Функциональное моделирование – определение и анализ функций и информационных потоков предметной области, например IDEF0.

Моделирование данных – на базе информационных потоков, выделяемых на предыдущем этапе разработки информационной модели предметной области.

Моделирование поведения – динамическое моделирование предметной области (IDEF3).

На этапе автоматической кодогенерации 3-х вышестоящих моделей выполняется генерация текстов программных компонентов.

На этапе сборки и квалификационных испытаний выполняется сборка и испытания результирующей системы, подсистемы или ПС.

Инкрементная модель

Как правило, инкрементные модели объединяют элементы каскадных моделей и прототипирования. Существуют различные варианты реализации инкрементных моделей. В классических вариантах модель основана на использовании полного, заранее сформированного набора требований, реализуемых последовательно в виде небольших проектов. Существуют также варианты модели, начинающиеся с формулирования общих требований. Требования постепенно уточняются в процессе разработки прототипов.

Каждая последующая версия прототипа добавляет новые функциональные возможности и может самостоятельно сдаваться в эксплуатацию. Вариант инкрементной модели жизненного цикла, предусматривающий изменение или уточнение требований на начальных этапах процесса разработки.

Достоинства:

1) возможность получения функционального продукта после реализации каждого инкремента;

2) короткая продолжительность создания инкремента; это приводит к ускорению начального графика поставки и снижению затраты поставки ПП;

3) предотвращение реализации громоздких перечней требований; возможность учета изменившихся требований;

4) снижение риска неудачи и изменения требований по сравнению с каскадной моделью;

5) включение в процесс пользователей, что позволяет оценить самые важные функциональные возможности продукта на более ранних этапах разработки и приводит к повышению качества ПП, снижению затрат и времени на его разработку;

6) стабильность требований во время созданияя определенного инкремента, возможности учета изменившихся требований.

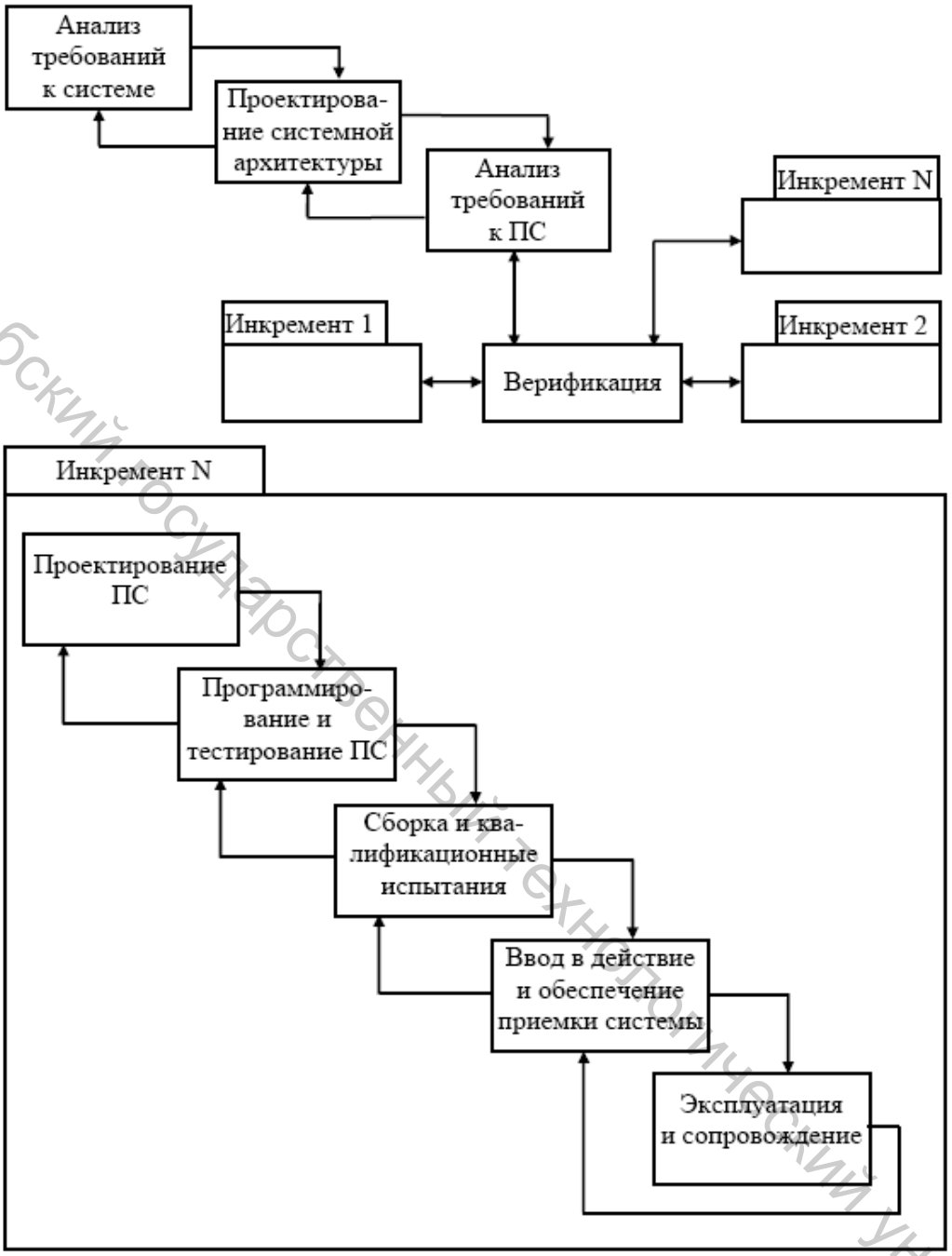

Рисунок 8 – Инкрементная модель

Недостатки:

1) необходимость полного функционального определения системы или программного средства в начале жизненного цикла для обеспечения определения инкрементов, планирования и управления проектом;

2) возможность текущего изменения требований к системе или программному средству, которые уже реализованы в предыдущих инкрементах;

3) необходимость хорошего планирования и проектирования, грамотного распределения работы;

4) наличие тенденции к оттягиванию решения трудных проблем на поздние инкременты, что может нарушить график работ.

Области применения:

1) при разработке проектов, в которых большинство требований можно сформулировать заранее, но часть из них может быть сформулирована через определенный период времени;

2) при необходимости быстро поставить на рынок продукт, имеющий функциональные базовые свойства;

3) для выполнения проектов с большим периодом разработки (один год и более);

4) при разработке проектов с низкой или средней степенью риска;

5) при выполнении проекта с применением новой технологии.

Вариант инкрементной модели по ГОСТ Р ИСО/МЭК ТО 15271-2002

Основана на использовании полного заранее сформированного набора требований и их постепенной реализации в отдельных инкрементах. Учитывает возможность как частичной параллельной разработки инкрементов, так и их последовательной разработки. Разработка каждого инкремента состоит из трех укрупненных этапов: проектирование, программирование и тестирование, ввод в действие и обеспечение приемки. Инкрементную модель можно комбинировать с другими моделями. Зачастую ее объединяют со спиральной или V-образной моделью, что позволяет снизить затраты и риски при разработке ПС (системы).

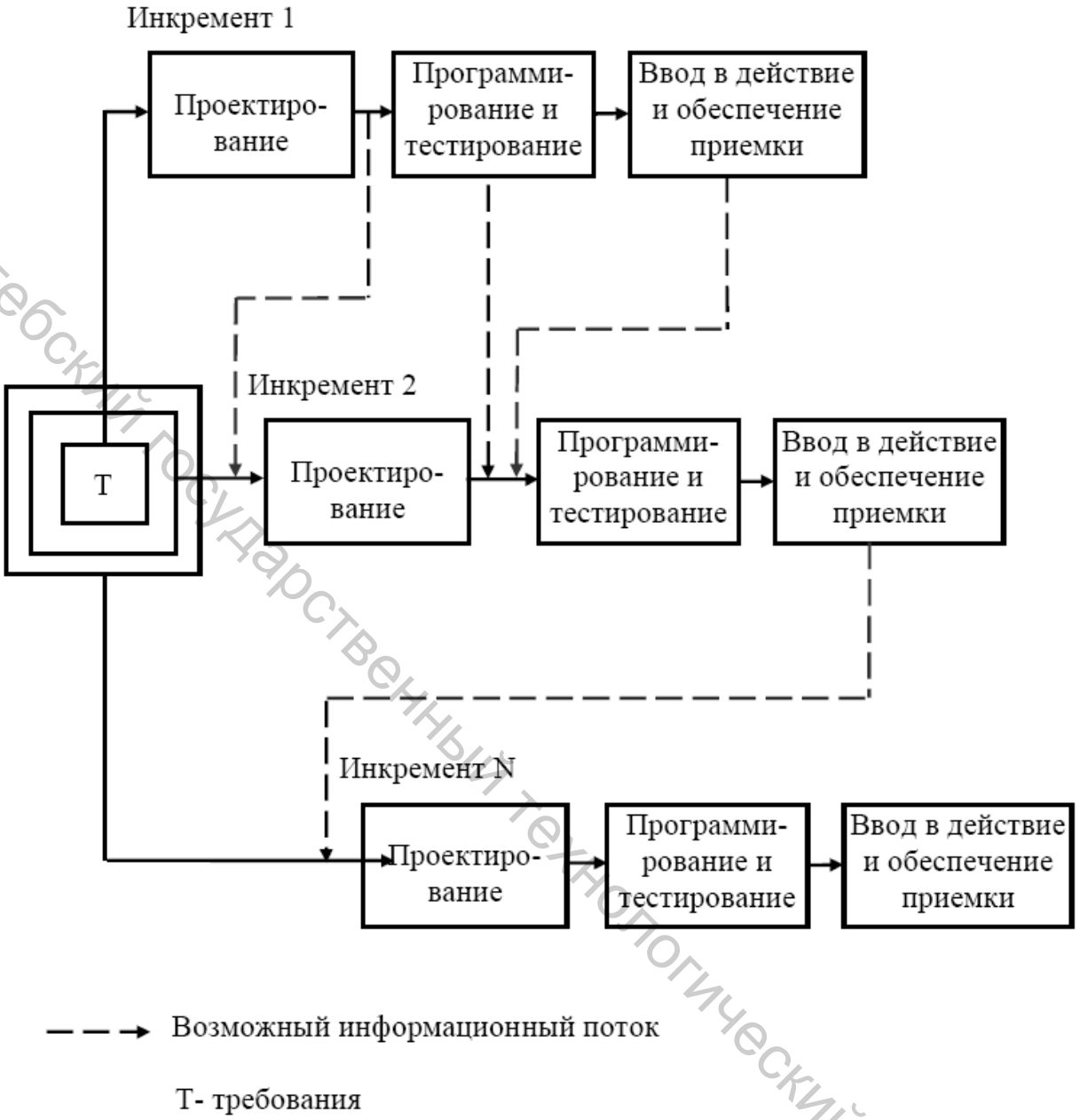

Рисунок 10 – Вариант инкрементной модели по ГОСТ Р ИСО/МЭК ТО 15271-2002

Эволюционная модель

В данном случае разработка каждой версии программного средства (системы) выполняется на основе каскадной модели, содержащей четыре этапа: разработка требований, проектирование, программирование и тестирование, ввод в действие и поддержка приемки. При разработке первой версии формулируются наиболее важные (базовые) требования к продукту. На основе данных требований разрабатывается и вводится в действие первая версия системы (программного средства). При разработке каждой очередной версии требования уточняются, при необходимости вводятся новые или

изменяются уже реализованные требования. На основе уточненных требований разрабатывается и вводится в действие очередная версия продукта. Процесс продолжается, пока все требования не будут окончательно уточнены и полностью реализованы.

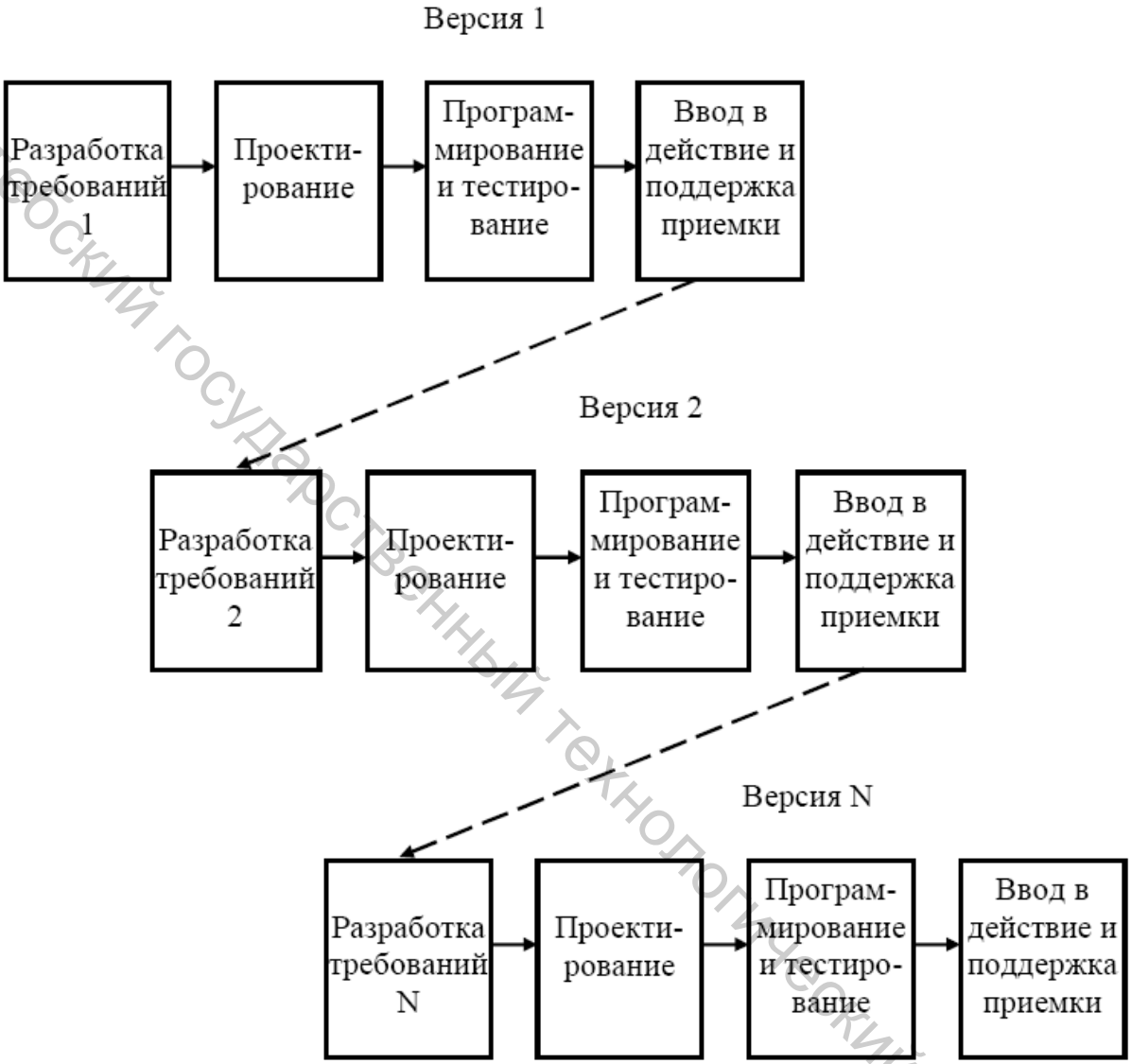

Рисунок 11 – Эволюционная модель ЖЦ ПС по ГОСТ Р ИСО/МЭК ТО 15271-2002

Достоинства:

1) возможность уточнения и внесения новых требований в процессе разработки;

2) пригодность для использования промежуточного продукта;

3) возможность обеспечения управления рисками;

4) реализация преимуществ инкрементной стратегии (выпуск версий, повторное использование результатов, сокращение графика работ);

5) обеспечение широкого участия пользователя в проекте, начиная с ранних этапов, что минимизирует возможность разногласий между заказчиками и разработчиками и обеспечивает создание продукта высокого качества;

6) уменьшение общих затрат на разработку за счет раннего решения проблем и меньшего объема доработок.

Недостатки:

1) сложность определения критериев для продолжения процесса разработки на следующей итерации, неизвестность точного количества необходимых итераций; это затрудняет планирование проекта и может вызвать задержку реализации конечной версии системы или программного средства;

2) сложность управления проектом; необходимость грамотного управления с целью ограничения количества итераций до разумного;

3) необходимость активного участия пользователей в проекте, что не всегда возможно в реальной жизни;

4) необходимость мощных инструментальных средств и методов прототипирования;

5) возможность отодвигания решения трудных проблем на последующие циклы, что может привести к несоответствию полученных продуктов требованиям заказчиков.

Области применения:

1) при разработке проектов, для которых требования слишком сложны, неизвестны заранее, непостоянны или их следует уточнить;

2) при разработке больших и долгосрочных проектов;

3) при разработке проектов используются новые технологии.

Структурная эволюционная модель быстрого прототипирования

При использовании структурной эволюционной модели быстрого прототипирования система или программное средство строится в виде последовательности прототипов.

Начало модели – в центре. На основном плана выполняется быстрый анализ требований, во время которых разрабатываются умышленно неполные требования. Затем на основе умышленных неполных требований производится укрупнение проектируемого прототипа, программирование и ввод в действие. Потом начинается цикл итераций быстрого прототипирования. Структура его аналогична структуре 1-го цикла. Далее, после того как пользователь оценил прототип и принял его, осуществляется детализированная разработка системы или ПС.

Отличие данной модели от других эволюционных моделей: с целью ускорения разработки результаты промежуточных циклов представляются в

виде прототипов, не доводятся до уровня рабочей версии и в эксплуатацию не сдаются. Назначение прототипа – уточнение требований.

Недостатки:

1) обычная недостаточность или неадекватность документации по ускоренным прототипам;

2) вероятность недостаточного качества результирующего ПС (или системы) за счет его создания из рабочего прототипа (при детальной разработке ПС или системы из последнего прототипа может оказаться сложной или невозможной реализация функций, не реализованных при итерационном прототипировании);

3) возможность задержки реализации конечной версии ПС (системы) при несочетании языка или среды прототипирования с рабочим языком или средой программирования.

Рисунок 12 – Структурная эволюционная модель быстрого прототипирования

Эволюционная модель прототипирования по ГОСТ Р ИСО/МЭК ТО 15271-2002.

Рисунок 13 – Эволюционная модель прототипирования по ГОСТ Р ИСО/МЭК ТО 15271-2002

Основой эффективного применения данной модели жизненного цикла является максимально возможная детализация на ранних этапах проекта (анализ требований к системе и проектирование системной архитектуры). Данные этапы выполняются в модели один раз. Однократное выполнение этих этапов достигается за счет тесных связей разработчиков с пользователями проекта. Требования к системе, в первую очередь, функции системы и внешние интерфейсы определяются пользователями в начале ЖЦ, деловые процессы уточняются при проведении пользователем серии оценок прототипов системы.

В данной модели при создании каждой версии ПС используется прототипирование. При разработке каждого прототипа уточняются требования к нему. Затем выполняется программирование прототипа в среде 4GL. При выполнении данного этапа инструментальная среда 4GL используется в первую очередь для быстрого проектирования и сборки ПС, а

также оперативного наращивания, изменения и уточнения ПС. Языки 4GL осуществляют частичную автоматическую кодогенерацию ПС.

Проверка и оценка каждого прототипа осуществляется пользователем в реальной эксплуатационной среде.

В модели ЖЦ определен фиксированный период проведения прототипирования и произвольное количество итераций.

Из описания данной модели видно, что при разработке прототипов фактически используется RAD-модель жизненного цикла.

Спиральная модель Боэма

Спиральная модель объединяет в себе преимущества других видов моделей, в нее включены анализ и управление рисками, процессы поддержки и управления, предусмотрены возможности использования прототипирования и быстрой разработки приложений. Каждый цикл разработки (итерация) представляет собой набор операций, соответствующий шагам в каскадной модели. Каждому витку спирали модели Боэма соответ фаза разработки. Каждая фаза может выполнятся за 1 или несколько циклов.

Рисунок 14 – Спиральная модель Боэма

А. ***Фаза разработки концепции*** (соответствует первому витку спирали).

1 – определение потребности;

2 – анализ рисков фазы разработки концепции;

3 – концептуальное прототипирование;

4 – разработка концепции требований к системе или программным средствам;

5– планирование проекта и процесса разработки.

B. ***Фаза анализа требований*** (соответствует второму витку спирали).

6 – анализ целей, альтернатив и ограничений, связанных с системой/программным средством;

7 – анализ рисков фазы анализа требований;

8 – демонстрационное прототипирование;

9 – оценка характеристик системы/продукта;

10 – разработка требований к системе/продукту;

11 – планирование перехода на фазу проектирования системы/продукта.

C. ***Фаза проектирования системы/продукта*** (соответствует 3 витку).

12 – анализ целей, альтернатив и ограничений, связанных с текущим циклом проектирования системы/продукта;

13 – анализ рисков фазы проектирования системы/продукта;

14 – оценочное прототипирование проектирования системы/продукта;

9 – см. выше;

15 – проектирование системной/программной архитектуры;

16 – планирование перехода на фазу технического проектирования ПС, программирования и сборки.

D. ***Фаза реализации (фаза технического проектирования, программирования и сборки)***, соответствует четвертому витку спирали).

17 – анализ возможности реализации в текущем цикле целей, альтернатив и ограничений, связанных с проектированием системы/продукта;

18 – анализ рисков фазы реализации;

19 – операционное прототипирование;

9 – см. выше;

20 – техническое проектирование ПС;

21 – программирование и тестирование ПС;

22 – сборка и квалификационные испытания ПС;

23 – сборка и квалификационные испытания системы;

24 – планирование перехода на фазу расширения функциональных возможностей, ввод в действие и обеспечение приемки версии.

E. ***Фаза сопровожденияия и расширения функциональных возможностей*** (соответствует 5 витку спирали).

25 – анализ целей, альтернатив и ограничений, связанных с сопровождением и поддержкой;

18, 19, 9, 20 – 23 – см. выше;

26 – приемочные испытания.

Оценочные действия:

27 – оценка концепции;

28 – оценка требований;

29 – оценка проектирования;

30 – оценка версии системы/продукта.

Действия, связанные с ***поставкой результатов*** текущего уровня разработки:

31 – поставка первой пригодной версии;

32 – поставка очередной пригодной версии;

33 – аудит конфигурации версии.

Модель поделена на **четыре квадранта**.

В квадранте I – анализ целей, альтернативных вариантов и ограничений – определяются рабочие характеристики, выполненные функциии, стабильность (возможности внесения изменений), аппаратно-программный интерфейс. Формулируются требования к продукту разработки данных фазы или цикла. Определяются альтернативные способы реализациии систем или ПП. Определяются ограничения, накладываемые на применение альтернативных вариантов, т. е. затраты, график выполнения, ограничение среды, здесь же определяются риски связанные с недостатком опыта в данной предметной области с применением новых технологий, жестким графиком, недостаток организационных процессов.

В квадранте II – оценка альтернативных вариантов, идентификация и разрешение рисков – выполняется оценка альтернативных вариантов, рассмотренных в предыдущем квадранте; оценка возможных вариантов разрешения рисков. Выполняется прототипирование как основа для работ следующего квадранта.

В квадрант III – разработка продукта текущего уровня – включаются действия по непосредственной разработке системы или программного продукта: проектирование системы и ее программных компонентов, разработка и тестирование исходных текстов программ, сборка, тестирование и квалификационные испытания продукта или системы и т. п.

В квадранте IV – планирование следующей фазы – выполняются действия, связанные с решением о переходе на цикл следующей фазы разработки или выполнении еще одного цикла текущей фазы разработки, в частности, оценка заказчиком результатов текущего уровня, разработка или коррекция планов проекта и следующего цикла, управление конфигурацией.

Программирование в данной модели выполняется значительно позже, чем в другой. Это позволяет минимизировать риски посредством последовательных уточнений требований, выдвигаемых пользователем. На каждой итерации рассматривается один или несколько главных факторов риска, начиная с фактора наивысшего риска.

Применение спиральной модели Боэма целесообразно при разработке проектов в организации, обладающей навыками адаптации модели с учетом его сложности и критичности.

Упрощенная спиральная модель ЖЦ ПС института качества SQI

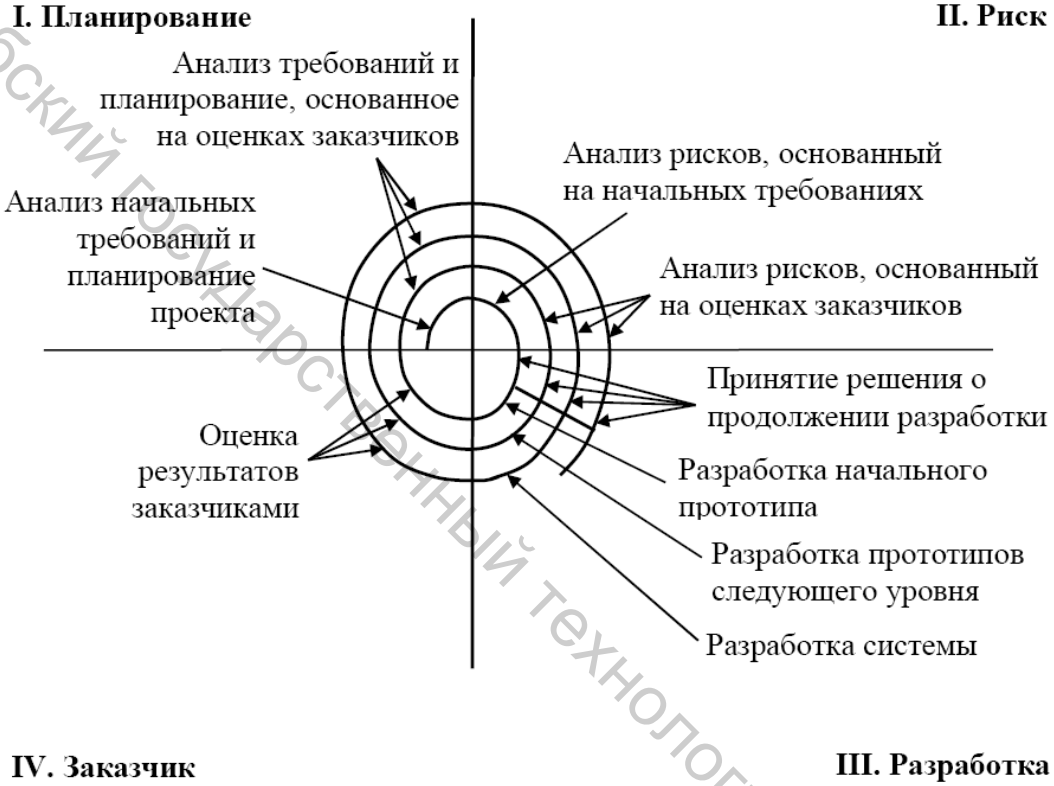

Рисунок 15 – Упрощенная спиральная модель ЖЦ ПС института качества SQI

В данной модели процесс жизненного цикла разработки проекта разделен на **четыре квадранта**: «Планирование», «Риск», «Разработка», «Заказчик». В пределах квадрантов выделяются только основные действия различного уровня.

Таким образом, в данной модели устранена чрезмерная детализация процесса. Необходимая детализация процессов предусматривается при выполнении работ этапов планирования конкретного цикла спирали.

Достоинства:

1) наличие действий по анализу рисков, что обеспечивает их сокращение и заблаговременное определение непреодолимых рисков;

2) усовершенствование административного управления процессом разработки, затратами, соблюдением графика и кадровым обеспечением, что достигается путем выполнения анализа в конце каждой итерации;

3) большое внимание в данных моделях уделяется действиям,

непосредственно не связанным с разработкой; это повышает качество как процесса разработки, так и продуктов разработки, упрощает прогнозирование сроков и стоимости разработки, повышает удовлетворенность заказчика результирующим продуктом;

4) возможность уточнения и внесения новых требований в процессе разработки новой версии;

5) пригодность для использования промежуточного продукта (очередной версии);

6) возможность обеспечения управления рисками;

7) обеспечение широкого участия пользователя в проекте, начиная с ранних этапов, что минимизирует возможность разногласий между заказчиками и разработчиками и обеспечивает создание продукта высокого качества;

8) уменьшение общих затрат на разработку за счет быстрого получения первой версии продукта.

Недостатки:

1) усложненность структуры модели, что приводит к сложности ее использования разработчиками, администраторами проекта и заказчиками; необходимость в высокопрофессиональных знаниях для оценки рисков;

2) высокая стоимость модели за счет стоимости и дополнительных временных затрат на планирование, определение целей, выполнение анализа рисков и прототипирование при прохождении каждого цикла спирали; неоправданно высокая стоимость модели для проектов, имеющих низкую степень риска или небольшие размеры;

3) необходимость активного участия пользователей в проекте, что не всегда возможно в реальной жизни;

4) необходимость мощных инструментальных средств и методов прототипирования.

Области применения:

1) при разработке проектов, для которых требования слишком сложны, неизвестны заранее, непостоянны или их следует уточнить;

2) при разработке больших и долгосрочных проектов;

3) при разработке проектов с использованием новых технологий.

Тема 3. Выбор модели жизненного цикла разработки программных средств и систем

Институтом качества программного обеспечения SQI (Software Quality Institute, США) специально для выбора модели жизненного цикла разработана схема классификации проектов по разработке программных средств и систем. Основу данной классификации составляют **четыре категории критериев**. По каждому из критериев проекты подразделяются на **два альтернативных класса**.

3.1 Категории критериев:

1) Характеристики требований к проекту.

Критерии данной категории классифицируют проекты в зависимости от требований пользователя (заказчика) к разрабатываемой системе или программному средству (свойств разрабатываемой системы или ПС).

2)Характеристики команды разработчиков.

Чтобы иметь возможность пользоваться критериями данной категории классификации проектов, состав команды разработчиков необходимо сформировать до выбора модели жизненного цикла.

Характеристики команды разработчиков играют важную роль при выборе модели жизненного цикла, поскольку разработчики несут ответственность за успешную реализацию проекта.

3) Характеристики пользователей (заказчиков).

Чтобы иметь возможность пользоваться критериями данной категории классификации проектов, до выбора модели жизненного цикла необходимо определить возможную степень участия пользователей (заказчиков) в процессе разработки и их взаимосвязь с командой разработчиков на протяжении проекта. Это важно, поскольку отдельные модели требуют усиленного участия пользователей в процессе разработки.

4) Характеристики типов проектов и рисков.

В некоторых моделях в достаточно высокой степени предусмотрено управление рисками. В других моделях управление рисками вообще не предусматривается.

Критерии данной категории отражают сложность проекта, достаточность ресурсов для его исполнения, учитывают график проекта и т.д. С учетом этого обеспечивается выбор модели, минимизирующей выявленные риски.

Следует отметить, что данная классификация проектов, направленная на обоснованный выбор модели жизненного цикла, применима для ***достаточно масштабных проектов*** по разработке программных средств и систем.

3.2 Процесс выбора модели ЖЦ разработки ПС и систем института SQI

Данный процесс базируется на применении таблиц вопросов. Каждый из вопросов предназначен для классификации анализируемого проекта по соответствующему критерию одной из четырех категорий классификации проектов.

Последовательность шагов процедуры:

1-й шаг процедуры. Проанализировать отличительные категории проекта по критериям, представленным в виде вопросов. Таблица 1 – таблица 4 отражают данные вопросы.

2-й шаг процедуры. Ответить на вопросы (см. таблица 1 – таблица 4) по анализируемому проекту, отметив слова «да» или «нет» в соответствующих

строках таблиц. Если слов «да» или «нет» в строке несколько, необходимо отметить все из них (все «да» или все «нет»).

3-й шаг процедуры. Расположить по степени важности категории (таблицы) или критерии, относящиеся к каждой категории (вопросы внутри таблицы), относительно проекта, для которого выбирается модель жизненного цикла.

4-й шаг процедуры. Выбрать из моделей (см. таблица 1 – таблица 4) ту модель, которая соответствует столбцу с наибольшим количеством отмеченных ответов с учетом их степени важности (с наибольшим количеством отмеченных ответов в верхней части приоритетных таблиц). Выбранная модель жизненного цикла является наиболее приемлемой для анализируемого проекта.

Таблица 1 – Выбор модели ЖЦ на основе характеристик требований

Критерий	Каскадная	V – образная	RAD	Инкре ментная	Эволю ционная	Спиральная
1 Являются ли требования к проекту легко определимыми и/или хорошо известными?	Да	Да	Да	Нет	Нет	Нет
2 Могут ли требования быть сформулированы в начале ЖЦ?	Да	Да	Да	Да	Нет	Нет
3 Будут ли изменяться требовавания на протяжении ЖЦ?	Нет	Нет	Нет	Нет	Да	Да
4 Нужно ли демонстрировать требования с целью их определения?	Нет	Нет	Да	Нет	Да	Да
5 Требуется ли проверка концепции?	Нет	Нет	Да	Нет	Да	Да
6 Будет ли формулирование требований зависеть от роста сложности системы в жизненном цикле?	Нет	Нет	Нет	Да	Да	Да

Таблица 2 – Выбор модели на основе характеристик команды разработчиков

Критерий	Каскадная	V – образная	RAD	Инкрементная	Эволюционная	Спиральная
1 Является ли проблемная предметная область проекта новой для болшинства разработчиков?	Нет	Нет	Нет	Нет	Да	Да
2 Являются ли технологии предметной области проекта новыми для большинства разработчиков?	Да	Да	Нет	Да	Нет	Да
3 Являются ли инструментальные средства в проекте новыми для болшинства разработчиков?	Да	Да	Нет	Нет	Нет	Да
4 Изменяются ли роли участников проекта на протяжении ЖЦ?	Нет	Нет	Нет	Да	Да	Да
5 Могут ли разработчики пройти обучение?	Нет	Да	Да	Да	Нет	Нет
6 Является ли структура разрабатываемых систем в жизненном цикле более значимой чем гибкость системы?	Да	Да	Нет	Да	Нет	Нет
7 Будет ли руководитель проекта отслеживать прогресс команды разработчиков в ходе проект?	Да	Да	Нет	Да	Нет	Да
8 Важна ли легкость распределения ресурсов проекта?	Да	Да	Да	Да	Нет	Нет
9 Приемлет ли команда разработчиклв оценки, проверки, стадии разработки?	Да	Да	Нет	Да	Да	Да

Таблица 3 – Выбор модели на основании характеристик коллектива пользователей

Критерий	Каскад	Vобр	RAD	Инкрементная	Эволюционная	Спиральная
1 Будет ли присутствие пользователя ограничено в ЖЦ разработки?	Да	Да	Нет	Да	Нет	Да
2 Будет ли пользователю представлено текущее состояние ПП?	Нет	Нет	Нет	Да	Да	Да
3 Будут ли пользователи знакомиться с проблемами предметной области в жизненном цикле разработки?	Нет	Нет	Да	Да	Да	Нет
4 Будут ли пользователи вовлечены в большинство фызы ЖЦ разработки?	Нет	Нет	Да	Нет	Да	Нет
5 Будет ли заказчик отслеживать ход выполнения проекта?	Нет	Нет	Нет	Нет	Да	Да

Таблица 4 – Выбор модели на основе характеристик типа проекта и рисков

Критерий	Каскад	Vобр	RAD	Инкрементная	Эволюционная	Спиральная
1 Будет ли проект идентифицировать новое направ продукта для организаци?	Нет	Нет	Нет	Да	Да	Да
2 Будет ли проект иметь тип системной интеграции?	Нет	Да	Да	Да	Да	Да
3 Будет ли проект являться расширением существующей системы?	Нет	Да	Да	Да	Нет	Нет
4 Будет ли проект крупным или среднемасштабным?	Нет	Нет	Нет	Да	Да	Да
5 Будет ли финансирование проекта стабильным на всем протяжении ЖЦ?	Да	Да	Да	Нет	Да	Нет
6 Будет ли продукт проекта критичным?	Нет	Нет	Нет	Нет	Да	Да
7 Ожидается ли длительная эксплуатация ПП в организации?	Да	Да	Нет	Да	Нет	Да
8 Необходим ли высокий уровень надежности продукта проекта?	Нет	Да	Нет	Да	Нет	Да
9 Предполагается ли эволюция ПП в течение жизненного цикла?	Нет	Нет	Нет	Да	Да	Да

Окончание таблицы 4

10 Будет ли система изменяться на этапе сопровождения?	Нет	Нет	Нет	Да	Да	Да
11 Является ли график сжатым?	Нет	Нет	Да	Да	Да	Да
12 Являются ли известными интерфейсные модули?	Да	Да	Нет	Да	Нет	Нет
13 Предполагается ли повторное использование компонентов?	Нет	Нет	Да	Да	Да	Да
14 Является ли достаточными ресурсы (время, деньги, персонал)?	Нет	Нет	Нет	Нет	Да	Да

3.3 Адаптация модели ЖЦ разработки ПС и систем к условиям конкретного проекта

Адаптация модели ЖЦ разработки ПС и систем к условиям конкретного проекта по СТБ ИСО/МЭК 12207-2003 и ГОСТ Р ИСО/МЭК ТО 15271-2002.

Характеристики проекта, влияющие на адаптацию.

В соответствии с данным стандартом выбор модели ЖЦ должен осуществляться при выполнениии 1-й работы (подготовка процесса разработки) процесса разработки. Выбор подходящ модели ЖЦ – это **первая стадия** применения модели в конкретном проекте.

Вторая стадия заключается в адаптации выбранной модели к потребностям данного проекта, к процессу разрабки, принятому в данной организации, и к требованиям действующих стандартов. С учетом этого должна быть выбрана и структура модели ЖЦПС работ и задачи процесса разработки из стандарта **СТБ ИСО/МЭК 12207-2003.** Данные работы и задачи могут пересекаться или взаимодействовать и выполняться рекурсивно.

К ***основным характеристикам проекта*** относятся:

- **организационные подходы** (процессы, связанные с защитой, безопасностью, конфиденциальностью, управлением риском, использованием независимого органа по верификации и аттестации, использованием конкретного языка программирования, обеспечением техническими ресурсами);
- **политика заказа** (например, типы договора);
- **политика сопровождения ПС** (ожидаемый период сопровождения и периодичность внесения изменений, критичность применения, персонал сопровождения и его квалификация, необходимая для сопровождения среда);
- **вовлеченные стороны** (например, заказчик, поставщик, разработчик, субподрядчик, посредники по верификации и аттестации, персонал сопровождения; численность сторон);
- **работы жизненного цикла системы** (подготовка проекта заказчиком, разработка и сопровождение поставщиком);

- **характеристики системного уровня** (количество подсистем и объектов конфигурации, межсистемные и внутрисистемные интерфейсы, интерфейсы пользователя, влияние ошибок ПС на защиту и безопасность системы, оценка временных мощностей и временных ограничений, наличие реализованных техническими средствами программ, наличие соответств компьютеров);
- **характеристики программного уровня** (количество программных объектов, типы, объемы и критичность программных продуктов, технические риски, типы документов, характеристики качества программных средств по **ISO/IEC 9126–1:2001** [3]); выделяются следующие ***типы программных продуктов***: 1) новая разработка; должны учитываться все требования к процессу разработки; 2) использование готового ПП; 3) модификация готового ПП; 4) ПП, встроенный или подключенный к системе; отдельно поставляемый и непоставляемый.

Объем проекта (в больших проектах, в которые вовлечены десятки или сотни лиц, необходим тщательный административный надзор и контроль с применением процессов совместного анализа, аудита, верификации, аттестации, обеспечения качества; для малых проектов такие методы контроля могут быть излишними);

Критичность проекта (значительная зависимость работы системы от правильного функционирования ПС и своевременности выдачи результатов; для таких ПС необходим более тщательный надзор и контроль).

Тема 4. Классические технологии разработки программных средств. Модульное проектирование программ

Под **модуляцией** (модуляризацией) понимается разделение программы на части по некоторым установленным правилам.

Модульная программа **–** это программа, в которой любую часть логической структуры можно изменить, не вызывая изменений в других частях программы.

4.1 Признаки модульности программы*:*

- Программа состоит из модулей.
- Независимость модулей. Это значит, что модуль можно изменять или модифицировать без последствий в других модулях.
- Условие «один вход – один выход». Модульная программа должна состоять из модулей, имеющих одну точку входа и одну точку выхода. В общем случае может быть и более чем один вход, но важно, чтобы точки входов были строго определены и другие модули не могли входить в данный модуль в произвольной точке.

Достоинства:

1. Упрощение разработки и реализации программ.
2. Облегчение чтения программ.

3. Упрощения настройки и модификации программ.
4. Облегчение работы с данными, имеющими сложную структуру.
5. Исключение чрезмерной детализации алгоритмов.

Недостатки:

Модульность требует большей дополнительной работы программиста и определенных навыков проектирования программ.

Модульный подход в среднем требует большего времени работы центрального процессора (в среднем на 5 – 10 %) за счет времени обращения к модулям.

Модульность программы приводит к увеличению объема программ (в среднем на 5 – 10 %).

4.2 Методы модуляции

Методы модуляции делятся на 3 группы:

- Методы нисходящего проектирования.
- Методы расширения ядра.
- Методы восходящего проектирования.

На практике обычно применяются различные сочетания этих методов.

Нисходящее проектирование программ

Основное **назначение** нисходящего проектирования – служить средством разбиения большой задачи на меньшие подзадачи так, чтобы каждую задачу можно было рассматривать независимо.

На начальном шаге в соответствии с назначением всей программы продумывается ее общая структура, без детальной проработки ее отдельных частей. Затем выделяются функции, которые будут востребованы на более низком уровне. В соответвие с ними разрабатываются отдельные компоненты, не детализованные на предыдущем шаге. Эти действия являются рекурсивные.

Стратегии, на которых основана реализия данного метода:

- пошаговое уточнение (данная стратегия разработана Е. Дейкстрой)
- анализ сообщений (данная стратегия базируется на работах группы авторов: Иодана, Константайна, Мейерса).

Эти стратегии отличаются способами определения начальных спецификаций требований, методами разбиения задачи на части и правилами записи (нотациями), положенными в основу проектирования.

При пошаговом уточнении на каждом следующем этапе декомпозиции детализируются программные компоненты очередного, более низкого уровня. При этом результаты каждого этапа являются уточнением результатов предыдущего этапа лишь с небольшими изменениями.

При пошаговом уточнении на каждом следующем этапе декомпозиции детализируются программные компоненты очередного более низкого уровня.

При этом результаты каждого этапа являются уточнением результатов предыдущего этапа лишь с небольшими изменениями.

Существуют различные способы реализации пошагового уточнения. Например:

1) кодирование программы с помощью псевдокода и управляющих конструкций структурного программирования;

2) использование комментариев для описания обработки данных.

Пошаговое уточнение требует, чтобы взаимное расположение строк программы обеспечивало читабельность всей программы.

Преимущества метода пошагового уточнения

Основное внимание при его использовании обращается на проектирование корректной структуры программ, а не на ее детализацию.

Так как каждый последующий этап является уточнением предыдущего лишь с небольшими изменениями, то легко может быть выполнена проверка корректности процесса разработки на всех этапах.

Недостаток метода пошагового уточнения*:* на поздних этапах проектирования может обнаружиться необходимость в структурных изменениях, требующих пересмотра более ранних решений.

Метод восходящего проектирования

При использовании метода восходящего проектирования в первую очередь определяются вспомогательные функции, которые могут потребоваться для проектирования программы. Эти функции реализуются с помощью модулей самых нижних уровн. Затем эти модули используются для определения функций более высокого уровня, и так до конца программы.

Недостаток – программисты начинают разрабатывать программу с несущественных, вспомогательных деталей, что затрудняет проектирование программы в целом.

Его использование целесообразно:

1) если существуют разработанные прикладные программы, которые могут быть использованы для выполнения некоторых функций всей разработки;

2) если заранее известно, что некоторые простые или стандартные модули потребуются нескольким различным частям программы (например, подпрограммы анализа ошибок, ввода-вывода и т. п.). Чаще всего используются сочетание восходящего и нисходящего методов.

Существуют 2 способа сочетания.

1 способ. Находятся ключевые (наиболее важные) модули промежуточных уровней, затем проектирование ведется нисходящим и восходящим методами одновременно.

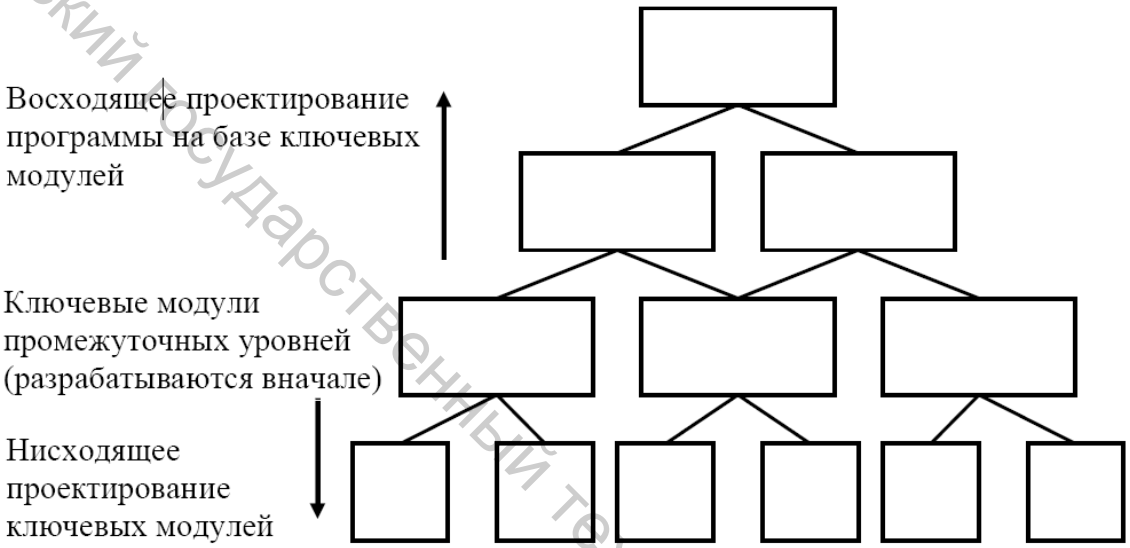

Рисунок 16 – 1 способ сочетания методов проектирования

2 способ. Проектируются модули нижнего уровня (т. е., которые необходимо спроектировать заранее). Затем программисты проектируют одновременно нисходящим и восходящим методами. При таком способе проектировании наиболее важной задачей является согласовать интерфейс между верхним и нижним уровнем программы, выполняемый в последнюю очередь. Это и есть недостаток.

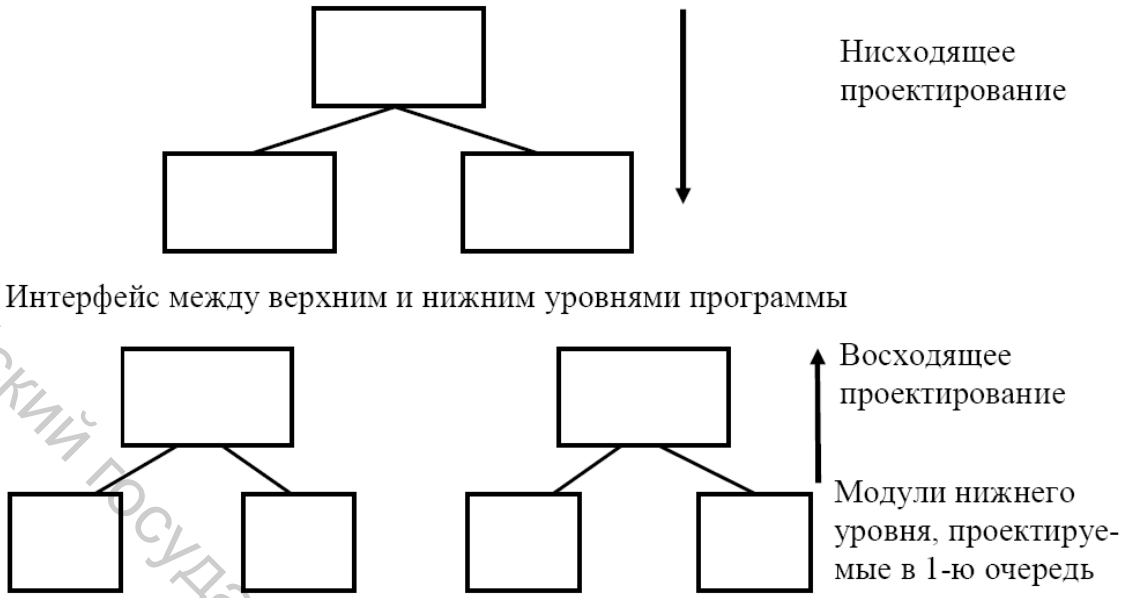

Рисунок 17 – 2 способ сочетания методов проектирования

Метод расширения ядра. Метод Джексона

В первую очередь проектируется основная часть программы – ядро, затем оно расширяется путем добавления функций. Метод разработан в 1975 г. Джексоном. Базируется на исходном положении, что структура программы зависит от формы подлежащей обработке данных. Поэтому структура данных может использоваться для фактического формирования структуры программы.

4.3 Оценка эффективности структурного разбиения программы на модули

Связность модуля. Типы и сила связности

Связность модуля определяется как мера независимости его частей. Чем выше связность модуля, тем больше отдельные части модуля зависят друг от друга и тем лучше результат проектирования.

Для количественной оценки связности используется понятие **силы связности модуля.** Типы связности модулей и соотвующие им силы связности представлены в таблице 5.

Таблица 5 – Типы связности модулей

Связность	Сила связности
Функциональная	10 (сильная связность)
Последовательная	9
Коммуникативная	7
Процедурная	5
Временная	3
Логическая	1
Связность по совпадению	0 (слабая связность)

Модуль с **функциональной связностью** выполняет единственную функцию и реализуется последовательностью операций в виде единого цикла. Пример – модуль, реализующий некоторый алгоритм-сортировку.

Модуль, имеющий **последовательную связность**, может быть разбит на последовательные части, выполняющие независимые функции, но совместно реализующие единственную функцию. Реализация как последовательность циклов (операций).

Модуль, имеющий **коммуникативную связность**, может быть разбит на независимые модули, разделяющие общую структуру данных.

Процедурная связность характерна для модуля, управляющие конструкции которого организованы так, как изображены на структурной схеме программы или схеме алгоритма, без выделения каких-либо функциональныхых частей.

Временная связность – модуль, содержащий функционально несвязанные части, которые необходимо обрабатывать в один и то же момент.

Логическая связность – если в модуле объединены операторы только по принципу их функционального подобия (все предназначены для проверки правильности данных). **Связность по совпадению** – если его операторы объединяются произвольным образом (например, по их непосредственному размещению в памяти).

Модули высшего уровня иерархической структуры программы должны иметь функциональную или последовательную связность. Для модулей обслуживания предпочтительна коммуникативная связность.

Если модули имеют процедурную, временную, логическую или связность по совпадению, то это говорит о недостаточном их планировании.

Сцепление модулей. Типы и меры сцепления

Сцепление модулей – это мера относительной независимости модулей.

Слабое сцепление определяется высоким уровнем независимости модулей.

Модули являются полностью независимыми, если каждый из них не содержит о другом никакой информации. Чем больше информации о другом модуле в них используется, тем менее они независимы и тем более сцеплены.

Таблица содержит типы сцепления модулей и соотвующие им степени сцепления.

Таблица 6 – Типы сцепления модулей

Сцепление	Степень сцепления
Независимое	0 (слабое сцепление)
По данным	1
По образцу	3
По общей области	4
По управлению	5
По внешним ссылкам	**7**
По кодам	9 (сильное сцепление)

Независимое сцепление возможно, если модули не вызывают друг друга и не обрабатывают одну и ту же информацию.

Модули **сцеплены по данным,** если они имеют общие простые элементы данных, которые передаются от одного модуля к другому как параметры (структура – массив, а передается в качестве параметра элемент массива; при этом изменения в структуре данных не повлияют на другой модуль). Модули со сцеплением по данным не имеют общей области данных (глобальных переменных).

Модули **сцеплены по образцу**, если в качестве параметров используются структуры данных (например, в качестве параметра передается массив). Недостаток: оба модуля должны содержать информацию о другом модуле (внутренней структуре данных), т. е. изменения должны отображаться и в другом модуле.

Модули **сцеплены по общей области,** если они разделяют одну и ту же глобальную структуру данных.

Модули **сцеплены по управлению**, если какой-либо из них управляет решениями внутри другого с помощью передачи флагов, переключателей и т. д.

Модуль **сцеплен по внешним ссылкам**, если у него есть доступ к данным другого модуля через внешнюю точку входа.

Модули **сцеплены по кодам**, если коды их команд объединены друг с другом, использующие общий участок памяти.

Если модули косвенно обращаются друг к другу (ч/з другие модули), то между ними также существует сцепление.

Сцепление модулей зависит от спроектированной структуры данных и способов взаимодействия между модулями.

Литература

1. Baecker, Ronald M. Human factors and tupography for more readable programs = Человеческие факторы и оформление для наиболее удобночитаемых програм / Baecker, Ronald M., Marcus Aaron. – Reading: Addison-Wesley publishing company, 1990. – 347 p. – (ACM Press). – Includes index. – ISBN 0-201-10745-7 : 180000-00.

2. ISO/IEC 12207:1995. Информационная технология – Процессы жизненного цикла программных средств.

3. ISO/IEC 15910:1999 – Информационная технология – Процесс создания документации пользователя программного средства.

4. ISO/IEC 9126–1:2001. Программная инженерия – Качество продукта.

5. Singh, S. K. Database Systems. Concepts, Design and Applications = Базы данных системы: концепции, дизайн и применение / S. K. Singh,. – New Delhi: Pearson, 2011. – 867 p. – Bibliography. – ISBN 978-81-7758-567-4 : 25000-00.

6. Аргила, К., Йордан, Э. Объектно-ориентированный анализ и проектирование систем. / Э.Йордан, К.Аргила. – М.: Лори, 2007

7. Бахтизин, В. В. Технология разработки программного обеспечения [Электронный ресурс]: учебное пособие для студентов высших учебных заведений по спец. «Программное обеспечение информационных технологий» / В. В. Бахтизин, Л. А. Глухова; УО «БГУИР». – Минск: БГУИР, 2010. – 1 CD-ROM (2,33 Мб): ил. – Систем. требования : Windows XP. В библиотеке нет печатного аналога. – Загл. с экрана. – ISBN 978-985-488-512-4 : 8000-00.

8. Брауде, Э. Технология разработки программного обеспечения. – СПб.: Питер, 2004.

9. Герман, О. В. Технология объектного программирования. Лабораторный практикум: учебно-методическое пособие в области информатики и радиоэлектроники для спец. 1-53 01 02 «Автоматизированные системы обработки информации» / О. В. Герман, А. В. Заяц. – Минск: БГУИР, 2013. – 78 с.: ил. – Спис. лит. – ISBN 978-985-488-843-9 : 15000-00.

10. Голенков, В. В. Интеллектуальный пользовательский интерфейс [Электронный ресурс]: учебно-методическое пособие для спец. 1-40 03 01 «Искусственный интеллект» / В. В. Голенков, Н. А. Гулякина, Д. Г. Колб; УО «БГУИР». – Минск: БГУИР, 2013. – 1 CD-ROM (2,06 Мб): ил. – Систем. требования : Windows XP. В библиотеке нет печатного аналога. – Загл. с экрана. – ISBN 978-985-488-953-5 : 8000-00.

11. ГОСТ Р ИСО/МЭК 12119–2000. Информационная технология. Пакеты программ. Требования к качеству и тестирование.

12. ГОСТ Р ИСО/МЭК 15910–2002. Информационная технология – Процесс создания документации пользователя программного средства.

13. ГОСТ Р ИСО/МЭК ТО 12182-2002 – Информационная технология. Классификация программных средств.

14. ГОСТ Р ИСО/МЭК ТО 15271–2002. Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207 (Процессы жизненного цикла программных средств).

15. Добрынин, В. Ю. Технологии компонентного программирования / В. Ю. Добрынин; Санкт-Петербургский университет. – Санкт-Петербург: Издательство Санкт-Петербургского университета, 2003. – 1 CD-ROM (1371 Кб).

16. Дубейковский, В.И. Эффективное моделирование с CA Erwin ProcessModeler (AllFusion) PM – М.: Диалог-МИФИ, 2009.

17. Ивасенко, А. Г. Информационные технологии в экономике и управлении: учебное пособие для студентов вузов, обучающихся по спец. «Прикладная информатика (по областям)», «Менеджмент организации», «Государственное и муниципальное управление» / А. Г. Ивасенко, А. Ю. Гридасов, В. А. Павленко. – 3-е изд., стер. – Москва: Кнорус, 2008. – 160 с.: рис. – Спис. лит. – ISBN 978-5-390-00103-5 : 14400-00.

18. Корпоративные информационные системы: пособие для студентов экономических спец. / Н. Н. Говядинова, Л. К. Голенда, Э. М. Дунько и др.; УО «БГЭУ» ; под общ. ред.: Л. К. Голенда, Н. Н. Говядиновой. – Минск, 2011. – 291 с.: ил. + Прил. – Лит. – ISBN 978-985-484-742-9 : 14774-00;14775-00.

19. Лаптев, В. В. C++. Объектно-ориентированное программирование: учебное пособие / В. В. Лаптев, – Санкт-Петербург: Питер, 2008. – 458 с.: ил. + Прил. – (Учебное пособие). – Спис. лит. – ISBN 978-5-91180-200-4 : 30736-00;29454-00.

20. Малыхина, М. П. Базы данных: основы, проектирование, использование: учебное пособие для студентов вузов, обучающихся по направлению подготовки «Информатика и вычислительная техника» / М. П. Малыхина, – Санкт-Петербург: БХВ-Петербург, 2006. – 528 с.: ил. – Спис. лит. – Предм. указ. – ISBN 5-94157-941-1 : 28680-00.

21. Материалы сайта www.microsoft.

22. Организация и использование баз данных в среде MS Access: методические указания к лабораторным работам для студентов механических спец. по предметам цикла «Информатика» / УО «ВГТУ» ; сост.: В. П. Терентьев, П. Г. Деркаченко. – Витебск, 2009. – 73 с.: табл. – Спис. лит. – Имеется электронный аналог. – 1550-00.

23. Оскерко, В. С. Технологии баз данных: учебное пособие для студентов учреждений, обеспечивающих получение высшего образования по экономическим спец. / В. С. Оскерко, З. В. Пунчик, О. А. Сосновский. – Минск: БГЭУ, 2007. – 171 с.: рис. – Спис. лит. – Глоссарий. – ISBN 978-985-484-406-0 : 6419-00;6420-00.

24. Программирование в среде Microsoft Visual Basic: лабораторный практикум по дисциплине «Информатика» для студентов спец. 1-37 01 01 «Двигатели внутреннего сгорания» / БНТУ ; сост. А. В. Предко. – Минск: БНТУ, 2014. – 56 с.: табл. – ISBN 978-985-550-348-5 : 20000-00.

25. Программирование сетевых приложений: лабораторные работы (практикум) для студентов спец. 1-40 01 01 «Программное обеспечение информационных технологий» и 1-40 01 02 «Информационные системы и технологии» / БНТУ ; сост. Н. М. Прибыльская. – Минск: БНТУ, 2013. – 66 с. – Лит. – ISBN 978-985-550-138-2 : 10000-00.

26. Проектная графика. Компьютерное проектирование: методические указания по курсу «Проектирование» для студентов спец. 1 19 01 01-01 / УО «ВГТУ» ; сост. Н. А. Абрамович. – Витебск, 2005. – 47 с.: рис. – Спис. лит. – 800-00.

27. СТБ ИСО/МЭК 12207-2003 – Информационная технология – Процессы жизненного цикла программных средств.

28. СТБ ИСО/МЭК 12207–2003. Информационная технология. Процессы жизненного цикла программных средств.

29. СТБ ИСО/МЭК 9126–2003. Информационные технологии. Оценка программной продукции. Характеристики качества и руководства по их применению.

30. Титоренко, Г.А. Информационные технологии управления: учебное пособие для студентов вузов, обучающихся по экономическим спец. – Москва: Юнити, 2008.

31. Фрост, Р. Проектирование и разработка баз данных. Визуальный подход = Database Design and Development / Р. Фрост, Д. Дей, Ван Слайк К. – Москва: NT Press, 2007. – 592 с.: ил. – Глоссарий. – Предм. указ. – ISBN 0-13-035122-9 (англ.) : 28560-00.

Учебное издание

Стасеня Тамара Петровна
Мандрик Ольга Геннадьевна

ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Курс лекций

Редактор *Т.П. Стасеня*
Технический редактор *Т.П. Стасеня*
Корректор *Н.В. Медведева*
Компьютерная верстка *М.В. Шпакова*

Подписано к печати 16.02.17. Формат 60x90 1/16. Усл. печ. листов 1,5.
Уч.-изд. листов 2.6.Тираж 60 экз. Зак. 73.

Учреждение образования «Витебский государственный технологический университет» 210035, г. Витебск, Московский пр-т, 72.

Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/172 от 12 февраля 2014 г.