Управление жизненным циклом информационных систем

СОДЕРЖАНИЕ

ЧАСТЬ II. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ЧАСТЬ I. КРАТКИЙ КОНСПЕКТ ЛЕКЦИЙ

1. ИНФОРМАЦИОННЫЕ СИСТЕМЫ

1.1. Виды и назначение информационных систем

Информационная система (ИС) – это совокупность программного обеспечения и электронного информационного хранилища (базы данных), разрабатываемая как единая система и предназначенная для автоматизации определённого рода деятельности. В литературе и стандартах ГОСТ информационные системы часто называют *автоматизированными системами*.

По роли, которую информационные системы играют в профессиональной деятельности, и решаемым ими задачам можно выделить следующие виды систем:

1) системы управления;
2) вычислительные информационные системы;
3) поисково-справочные информационные системы;
4) системы поддержки принятия решений;
5) информационные обучающие системы.

Функции (функциональные возможности) любой информационной системы можно разбить на следующие классы:

1) функции редактирования данных;
2) функции получения информации из информационного хранилища (поисковые функции);
3) функции безопасности (управления доступом);
4) расчётные функции;
5) технологические функции (автоматизация деятельности).

Современные информационные системы могут быть классифицированы по различным признакам. Рассмотрим наиболее важные из них.

1. *Классификация по масштабу.* По масштабу (количеству и сложности решаемых задач и выполняемых функций) ИС делятся на однопользовательские, групповые и корпоративные.

2. *Классификация по характеру использования информации.* Среди всего многообразия видов ИС можно особо выделить два класса систем: информационно-поисковые и управляющие.

3. *Классификация по степени автоматизации.* В зависимости от того, в какой степени автоматизировано использование пользователем функций системы, выделяют ручные, автоматизированные и автоматические ИС.

1.2. Архитектура информационной системы

Информационная система представляет собой совокупность базы данных (БД) и программно-аппаратных средств, которые совместно обеспечивают хранение, поиск и обработку информации, а также реализуют методы взаимодействия с пользователем.

Архитектура ИС – концепция, определяющая модель системы, структуру, выполняемые ею функции и взаимосвязи между её компонентами, слоями и звеньями.

По выполняемым функциям компоненты архитектуры ИС можно разделить на три связанных друг с другом слоя: слой представления, слой бизнес-логики и слой доступа к данным.

1. *Слой представления* представляет собой реализацию пользовательского интерфейса, в том числе – функций для организации взаимодействия с пользователем: работа с окнами и формами, нажатие кнопок, движение мыши, построение изображений, ввод данных в формах ввода, вывод данных в форме таблиц и т.д. Функции слоя представления отвечают только за визуальное представление и форматирование данных и не выполняют анализ и обработку данных, включая поиск, отбор, сортировку и т.д. Функции слоя представления оперируют понятиями и элементами пользовательского интерфейса ИС.

2. *Слой бизнес-логики* содержит правила и алгоритмы реакции приложения на действия пользователя или на внутренние события, правила анализа и обработки данных. Функции слоя решают задачи управления данными на абстрактном уровне, в общем виде, без привязки к конкретной реализации пользовательского интерфейса и к конкретному способу хранения данных ИС. Функции слоя бизнес-логики оперируют концептуальными сущностями – объектами предметной области.

3. *Слой доступа к данным* отвечает за создание, хранение, выборку, модификацию и удаление данных, связанных с решаемой прикладной задачей. На данном слое реализуются функции, обеспечивающие взаимодействие с базой данных, с учётом специфики конкретной системы управления базами данных (СУБД) и выбранных технологий доступа к данным, используемых при разработке программного обеспечения (ПО) информационной системы. Слой доступа к данным обеспечивает переход от логической структуры реляционной базы данных (таблиц) к концептуальным сущностям – объектам предметной области.

По физическому размещению компонентов информационной системы и логическому распределению их функционала на серверах и рабочих станциях различают звенья архитектуры информационной системы. В настоящее время при создании ИС широко используется двухзвенная архитектура типа «*клиент-сервер*».

Сервером является компьютер, на котором установлена и постоянно функционирует СУБД, которая обеспечивает непрерывный доступ к базе данных ИС с любого узла локальной сети. На сервере ИС реализован слой доступа к данным.

Клиентом является компьютер (рабочая станция) пользователя, на котором выполняются прикладные программы. Клиентские приложения реализуют функции слоя представления (пользовательский интерфейс, ввод и вывод данных) и слоя бизнес-логики (операции анализа и обработки данных, формирование отчётов).

Вся бизнес-логика и все механизмы формирования пользовательского интерфейса реализованы на стороне клиента с помощью клиентского приложения, установленного на компьютере пользователя. Сервер используется только для работы СУБД и физического хранения файлов БД.

Недостатком описанной реализации клиент-серверной архитектуры является то, что сервер фактически рассматривается только как удалённая база данных, а экземпляры клиентских приложений, реализующие бизнес-логику анализа и обработки данных, выполняются на каждом отдельном компьютере-клиенте. Следствием этого является высокая нагрузка на локаль-

ную вычислительную сеть (ЛВС), так как для анализа и обработки данных между клиентом и сервером БД необходимо перемещать большой объём данных.

Для сокращения нагрузки на ЛВС и централизации анализа и обработки данных рекомендуется использовать другой способ реализации клиент-серверной архитектуры, который предполагает перенос реализации слоя бизнес-логики и слоя представления на отдельный сервер ИС – *сервер приложений*.

В общем случае сервер слоёв бизнес-логики и представления и сервер базы данных может представлять собой два разных сервера. В таком случае происходит переход от двухзвенной архитектуры к трёхзвенной архитектуре, которая включает в себя:

1) клиентское приложение на рабочей станции конечного пользователя;
2) сервер приложений, на котором реализованы слой бизнес-логики и слой представления;
3) сервер базы данных, который обеспечивает функционирование СУБД и физическое хранение данных информационной системы.

По сравнению с двухзвенной архитектурой трёхзвенная архитектура объединяет действия по формированию пользовательского интерфейса, анализу и обработки данных, формированию отчётов не в клиентском приложении, выполняющемся на рабочей станции пользователя, а в приложении, выполняющемся на сервере приложений. При этом сервер БД по-прежнему решает только задачи управления информацией, хранящейся в базе данных, а также обеспечивает физическое хранение файлов ИС.

Трёхзвенная архитектура является эталонной с точки зрения компонентного физического распределения двух различных функциональных компонентов системы (сервер приложений и сервер базы данных) по разным технологическим узлам (серверам локальной вычислительной сети). При отсутствии возможности реализовать архитектуру системы в таком виде допускается объединение функций сервера приложений и сервера базы данных на одном сервере.

2. МОДЕЛИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

2.1. Язык моделирования *UML*

Для описания устройства и поведения любой сложной системы приходится разрабатывать большое количество взаимосвязанных моделей. В применении к разработке ИС это означает, что необходим универсальный язык моделирования, с помощью которого можно описывать различные аспекты архитектуры и поведения системы на каждом из этапов её жизненного цикла (ЖЦ), от анализа требований до внедрения. В настоящее время одним из наиболее популярных языков моделирования, применяющихся в процессе разработки ПО, является унифицированный язык моделирования *Unified Modeling Language* (*UML).*

Унифицированный язык моделирования *UML* – это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех компонентов, создаваемых при разработке программных систем.

Язык *UML* обладает следующими характеристиками:

1) обеспечивает поддержку всех этапов ЖЦ ПО;
2) является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей;
3) содержит механизмы расширения и специализации базовых концепций языка;
4) является одной из стандартных нотаций визуального моделирования программных систем и поддерживается многими объектно-ориентированными *CASE*-продуктами.

Словарь языка *UML* включает три вида строительных блоков: сущности, отношения и диаграммы. В языке *UML* имеется четыре вида сущностей (структурные, поведенческие, группирующие, аннотационные), четыре вида отношений (зависимость, ассоциация, обобщение, реализация) и восемь типов диаграмм.

Диаграмма UML – это графическое представление набора элементов, изображаемое в виде связанного графа с вершинами (сущностями) и ребрами (отношениями), используемое для визуализации системы с разных точек зрения.

Наиболее часто для построения моделей ИС используются следующие виды диаграмм языка UML: диаграммы прецедентов, диаграммы деятельности, диаграммы взаимодействия (диаграммы последовательности или кооперативные диаграммы), диаграммы состояний, диаграммы классов, диаграммы базы данных, диаграммы компонентов, диаграммы развертывания.

Обычно, за исключением самых простых моделей, диаграммы дают свернутое представление элементов, из которых состоит разрабатываемая система. Один и тот же элемент системы может присутствовать во всех диаграммах или только в нескольких.

Диаграммы *UML* – это средство визуализации модели разрабатываемой программной системы. С помощью диаграмм можно описать систему с различных точек зрения. При этом ни одна из диаграмм по отдельности не является достаточной для полного описания системы, поскольку каждая диаграмма фокусируется на каком-то определенном аспекте функционирования системы и описывает его на определённом уровне абстракции.

Каждая диаграмма соответствует некоторой частной точке зрения на разрабатываемую систему. Ни одна отдельно взятая диаграмма не может являться моделью системы. Только набор нескольких диаграмм разного вида может выступать в качестве модели системы и относительно полно ее описывать.

2.2. Применение языка *UML* при создании ИС

Диаграммы языка *UML* используются для описания различных аспектов функционирования и структуры ИС на разных стадиях создания системы и, соответственно, на разных этапах моделирования: концептуального, логического и физического.

На этапе создания *концептуальной модели* для описания бизнес-процессов используются модели прецедентов и диаграммы деятельности, а для описания бизнес-объектов – модели бизнес-объектов и диаграммы последовательности.

На этапе создания *логической модели* ИС описание требований к системе задается в виде модели системных прецедентов, а предварительное проектирование осуществляется с использованием диаграмм классов, диаграмм последовательностей и диаграмм состояний.

На этапе создания *физической модели* детальное проектирование выполняется с использованием диаграмм классов, диаграмм компонентов и диаграмм развертывания.

Рассмотрим последовательность этапов создания ИС и использование диаграмм языка *UML* для построения моделей системы.

1. Разработка модели прецедентов. Проектирование ИС начинается с изучения и моделирования деятельности организации. Модель прецедентов описывает бизнес-процессы с точки зрения внешнего пользователя. Этап завершается после разработки диаграмм деятельности для всех выделенных прецедентов. На последующих этапах анализа и проектирования ИС могут быть выявлены дополнительные подробности деятельности объекта автоматизации. Поэтому разработанная модель прецедентов может в дальнейшем неоднократно корректироваться.

2. Разработка модели бизнес-объектов. Следующим этапом проектирования ИС является разработка модели бизнес-объектов, которая описывает выполнение бизнес-процессов организации ее внутренними исполнителями. Этап завершается после разработки необходимого количества диаграмм последовательности. Результатом этапа являются согласованные с заказчиком и достаточно подробные описания действий специалистов организации, внедряющей ИС, необходимые для обеспечения исполнения её функций.

3. Разработка концептуальной модели данных. На основе информации, полученной на предыдущих этапах, выполняется разработка концептуальной модели данных, которые будут использоваться в разрабатываемой системе. Разработанные диаграммы являются отправной точкой в процессах проектирования базы данных и приложений ИС, обеспечивают согласованность действий бизнес-аналитиков и разработчиков в процессе дальнейшей работы над системой.

4. Разработка требований к системе. На этапе формирования требований определяется область действия разрабатываемой системы и формируется представление о желаемых возможностях разрабатываемой ИС. Основой разработки требований является модель системных прецедентов, отражающая выполнение конкретных обязанностей внутренними и внешними исполнителями с использованием ИС. Результатом выполнения этапа является не только исчерпывающий перечень функций, которые должны быть реализованы в проектируемой системе, но и подробное описание необходимой реализации этих функций.

5. Анализ требований и предварительное проектирование системы. Основной задачей этапа является разработка общего для всех участников разработки предварительного проекта ИС, отвечающего всем ранее сформулированным требованиям. На основе имеющейся модели системных прецедентов строятся диаграммы классов. В результате выполнения этапа появляется достаточно подробное описание состава и функций проектируемой системы, а также используемых в работе ИС данных.

6. Разработка моделей базы данных и приложений. На этом этапе осуществляется отображение элементов полученных ранее моделей классов в элементы моделей базы данных и приложений (классы отображаются в таблицы БД, атрибуты классов – в столбцы и т.д.). Поскольку модели базы данных и приложений строятся на основе единой логической модели, автоматически обеспечивается связность этих проектов. Результатом этапа является детальное описание проекта базы данных и приложений системы.

7. Проектирование физической реализации системы. На этом этапе модели БД и приложений ИС дополняются описанием их размещения на технических средствах разрабатываемой системы. Диаграммы развертывания позволяют отобразить на единой схеме различные компоненты системы (программные и информационные) и их распределение по комплексу технических средств.

Таким образом, диаграммы языка *UML* могут создаваться и использоваться в качестве моделей ИС на всех стадиях её разработки, на всех этапах жизненного цикла системы.

3. ЖИЗНЕННЫЙ ЦИКЛ ИНФОРМАЦИОННЫХ СИСТЕМ

3.1. Модели жизненного цикла

Разработка информационной системы является весьма сложной задачей, процесс решения которой разбивается на определённое количество этапов. К числу ключевых этапов относятся: анализ требований, проектирование, реализация, тестирование и внедрение. Объединение этих этапов в один процесс приводит к понятию жизненного цикла ИС.

Жизненный цикл (ЖЦ) информационной системы – непрерывный процесс, который начинается с момента принятия решения о необходимости создания системы и заканчивается в момент её полного изъятия из эксплуатации.

Модель жизненного цикла ИС – структура, описывающая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного обеспечения в течение всей жизни ИС, от определения требований до завершения её использования.

В дальнейшем, рассматривая различные модели ЖЦ, для простоты будем использовать следующий набор этапов:

1. Анализ (разработка требований).
2. Проектирование (создание проекта).
3. Реализация (программирование).
4. Тестирование (исправление ошибок).
5. Внедрение (ввод в эксплуатацию).

К настоящему времени наибольшее распространение получили следующие основные модели ЖЦ:

1) каскадная (водопадная) модель и её варианты;

2) инкрементная модель;

3) спиральная модель.

Модель ЖЦ является методологической основой для организации процесса разработки реальной ИС.

3.2. Каскадная модель

Каскадная или *водопадная* модель ЖЦ является классической моделью однократного прохода, которая описывает линейную последовательность этапов создания ИС (рис. 3.1).

Каскадная модель ЖЦ предусматривает выполнение стадий жизненного цикла в строго определённом порядке. Переход на следующую стадию осуществляется только после полного завершения работ на предыдущей стадии. Данная модель детально описана в ГОСТ 34.601–90.

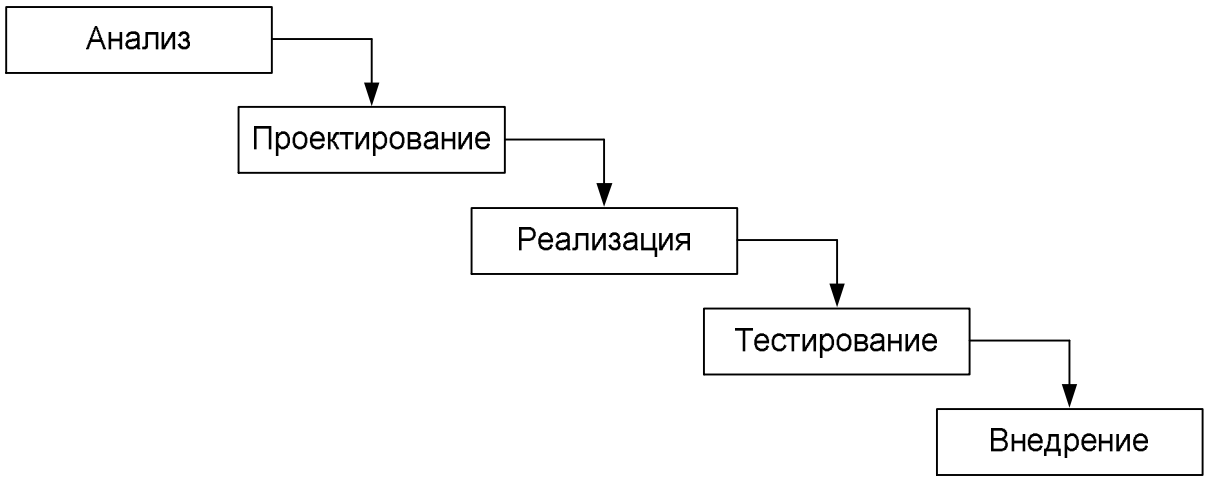

Рис. 3.1. Каскадная (водопадная) модель жизненного цикла ИС

Достоинством каскадной модели является явное описание всех этапов работы и определение последовательности их реализации. Это позволяет планировать сроки завершения работ и соответствующие затраты.

Недостатком каскадной модели является то, что реальный процесс создания ИС в действительности практически никогда не укладывается в жёсткую каскадную схему. Постоянно возникает потребность в возврате к предыдущим этапам для уточнения требований и исходных данных.

Каскадная модель с промежуточным контролем является модификацией каскадной модели ЖЦ, которая по окончании текущего этапа предусматривает возможность перехода на предыдущий этап для уточнения требований.

Межэтапные корректировки позволяют учитывать и сглаживать ошибки результатов выполнения предыдущих этапов.

Этот подход частично снимает недостатки классической каскадной модели.

V-образная каскадная модель ЖЦ является ещё одним способом усовершенствовать классическую модель. Её отличает то, что каждому шагу этапов анализа, проектирования и реализации соответствует отдельный шаг на этапах тестирования и внедрения.

Общим недостатком всех каскадных моделей ЖЦ является то, что для них требования к ИС зафиксированы в виде формальной спецификации и не могут быть изменены в процессе создания системы. Таким образом, заказчик зачастую получает систему, не соответствующую его ожиданиям.

3.3. Инкрементная модель

Инкрементная модель ЖЦ отличается от классической каскадной тем, что в ней существует сразу несколько комплектов требований к системе (спецификаций) с разной степенью полноты. Вся разработка делится на заданное количество шагов (итераций, инкрементов). В процессе разработки под каждый набор требований создаётся своя версия информационной системы. Таким образом, результатом разработки является не одна, а несколько версий ИС, создаваемых последовательно друг за другом.

При использовании инкрементной модели ЖЦ обычно особо выделяют базовый набор требований к ИС, который определяет функциональные возможности первой версии системы – её прототипа.

Прототип – версия ИС, предназначенная для демонстрации заказчику некоторых ключевых свойств будущего продукта. Создание прототипа позволяет вовлечь заказчика в разработку информационной системы в самом начале работы.

Результатом выполнения последнего шага является окончательная версия ИС, готовая к вводу в эксплуатацию.

Главным достоинством инкрементной модели ЖЦ является то, что такой жизненный цикл позволяет заказчику контролировать процесс разработки системы, начиная с её самой ранней версии – прототипа.

Недостатком инкрементной модели является то, что, как и для классической каскадной модели ЖЦ, перед началом разработки необходимо сформулировать полный набор требований к информационной системе для каждой версии, включая прототип и промежуточные версии.

3.4. Спиральная модель

Одной из наиболее эффективных подходов к разработке сложных ИС является использование эволюционной стратегии разработки. В этом случае система строится в виде последовательности версий, причём в начале процесса определены не все требования. В процессе разработки требования уточняются, и система непрерывно дорабатывается.

Спиральная модель ЖЦ относится к эволюционным моделям (рис. 3.2). Каждый виток раскручивающейся спирали соответствует разработке одной (начальной, промежуточной или окончательной) версии ИС и представляет собой полный цикл разработки, начиная с анализа и заканчивая внедрением.

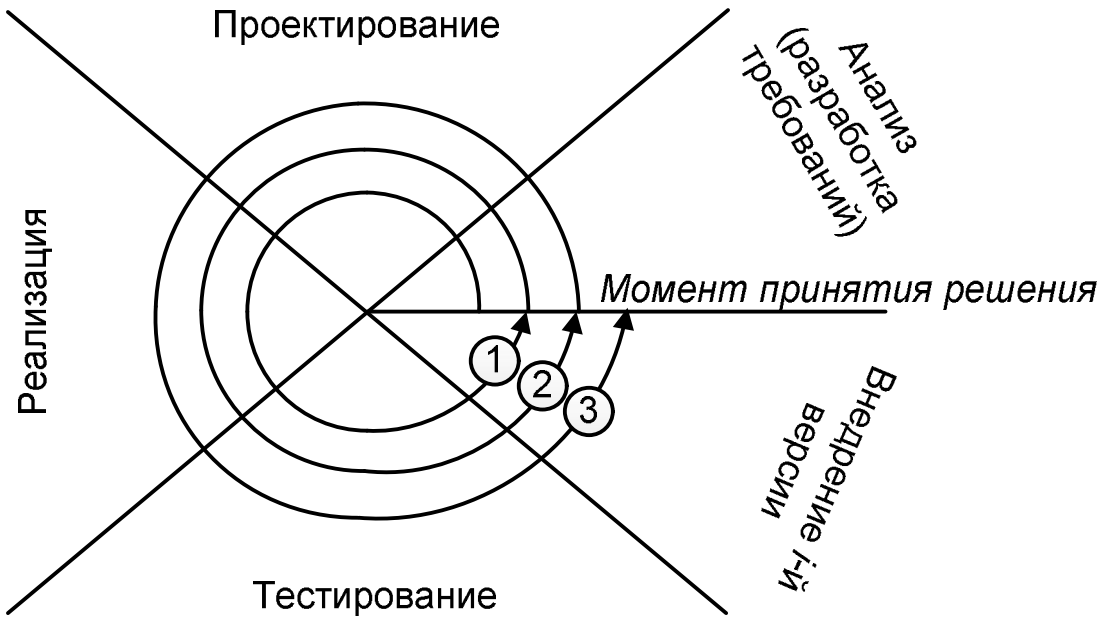

Рис. 3.2. Спиральная модель («1», «2», «3» – номера версий)

Спиральная модель отличается от инкрементной модели тем, что первый этап каждой итерации (анализ и разработка требований) выполняется только после завершения предыдущей

итерации и выпуска очередной версии системы. Причём этот анализ проводится с учётом полученных результатов и только после согласования этих результатов с заказчиком. Таким образом, нет необходимости заранее выполнять анализ и формулировать требования для всех итераций.

Другим важным отличием спиральной модели ЖЦ является то, что количество требуемых итераций заранее неизвестно. Перед началом каждой итерации выполняется анализ полученных результатов и принимается решение о продолжении или прекращении разработки системы. Если цель достигнута, и разработанная система полностью удовлетворяет потребностям заказчика, то разработка прекращается. Если же возникает необходимость доработки информационной системы, то процесс разработки переходит на следующий виток спирали.

Достоинством спиральной модели ЖЦ является то, что до реализации доводится обоснованный окончательный вариант ИС, который удовлетворяет действительным требованиям заказчика. Таким образом, снижаются риски, связанные с неправильным пониманием потребностей заказчика или неправильной реализацией требований к системе.

Другим достоинством спиральной модели жизненного цикла является ускорение разработки ИС, обусловленное более активным привлечением заказчика к формированию требований на основе анализа работы промежуточных версий.

Главный недостаток спиральной модели – сложность планирования работ и оценки затрат, сроков и рисков выполнения проекта. Основной проблемой является определение момента перехода на следующую итерацию. Для её решения вводятся ограничения на длительность этапов и итераций по времени.

4. СОВРЕМЕННЫЕ МЕТОДОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

4.1. Методология *Microsoft Solutions Framework*

Методология разработки ПО *Microsoft Solutions Framework* (*MSF*) разработана компанией *Microsoft* на основе своего практического опыта работы над программными проектами. Методология описывает подходы и принципы управления людьми и рабочими процессами для организации процесса разработки программного обеспечения.

Методология *MSF* представляет собой согласованный набор концепций, моделей и правил, определяющих процесс разработки ПО. Руководство по *MSF* структурно состоит из пяти документов, каждый из которых описывает определённую *модель* или *дисциплину*: модель процессов, модель проектной группы, дисциплину управления проектами, дисциплину управления рисками, дисциплину управления подготовкой.

В контексте изучения моделей жизненного цикла ИС наибольший интерес для анализа представляет *модель процессов MSF*, которая сочетает в себе свойства двух основных моделей ЖЦ: каскадной и спиральной.

От спиральной модели ЖЦ методология *MSF* использует подход, основанный на итеративной разработке. Весь ЖЦ проекта протекает в виде последовательности итераций, каждая из которых заканчивается выпуском версии. Методология *MSF* рекомендует вкладывать в первую версию продукта только базовую функциональность и затем наращивать ее в следующих версиях. Как и в обычной спиральной модели ЖЦ в модели процессов *MSF* пересмотр функциональности, планов, спецификаций и требований не прекращается до конца проекта и производится после каждой итерации. Такой подход позволяет планировать после-

дующие итерации, учитывая опыт предыдущих, обеспечивая гибкость и устойчивость к изменению требований заказчика.

Элементы каскадной модели ЖЦ реализуются в модели процессов *MSF* в виде системы *вех* и *фаз*.

Вехи – это контрольные точки проекта, характеризующие достижение в его рамках какого-либо существенного (промежуточного или конечного) результата. В отличие от этапа или стадии, которые описывают характер и объём работ, вехи определяют цели разработки. Методология *MSF* предусматривает следующие ключевые вехи:

1. Концепция проекта утверждена.
2. Планы проекта утверждены.
3. Разработка завершена.
4. Готовность решения утверждена.
5. Внедрение завершено.

Кроме этого может существовать большое количество промежуточных вех, которые показывают достижение в ходе проекта определенного прогресса и расчленяют большие сегменты работы на меньшие, обозримые участки.

Фазы – это этапы (стадии) между вехами. Модель процессов *MSF* включает следующие основные фазы процесса разработки: выработки концепции, планирования, разработки, стабилизации и внедрения.

Фазы в модели процессов *MSF* по их последовательности, характеру и задачам соответствуют этапам в стандартных каскадной и спиральной моделях ЖЦ. Главной особенностью модели процессов *MSF* является наличие вех – контрольных точек, позволяющих качественно и количественно оценить результат выполнения работ. Таким образом, процесс разработки программного продукта становится более формализованным и управляемым.

4.2. Методология *Rational Unified Process*

Методология разработки ПО *Rational Unified Process* (*RUP*) разработана компанией *Rational Software*. Методология описывает, как эффективно применять коммерчески обоснованные и практически опробованные подходы к разработке программных

продуктов. Методология *RUP* включает в себя комплекс руководств, примеров, шаблонов и наставлений по использованию инструментальных средств для выполнения всех возможных работ по созданию и сопровождению программного продукта.

Особенностью *RUP* является то, что вместо написания большого количества текстовых документов в течение всего процесса разработки создаются, корректируются и активно используются графические модели, описывающие различные стороны разрабатываемого программного продукта. Эти графические модели формируют общую базу знаний, доступную каждому члену группы разработчиков. В качестве языка моделирования в общей базе знаний используется унифицированный язык моделирования *UML*.

С точки зрения организации процесса разработки методология *RUP* использует итеративную модель ЖЦ. Процесс разработки состоит из четырёх фаз, каждая из которых включает в себя одну или несколько итераций. Количество итераций, как и в других процессах разработки и моделях ЖЦ, определяется сложностью создаваемой системы.

В отличие от спиральной модели ЖЦ, методология *RUP* использует фазы для группировки отдельных итераций с точки зрения достигаемых результатов:

1. *Начальная фаза*: общее описание системы (основные требования, характеристики и ограничения), план проекта.
2. *Фаза уточнения*: функциональные требования, архитектура системы (модель предметной области, технологическая платформа), проект системы, прототип системы.
3. *Фаза конструирования*: продукт, готовый к внедрению (программное обеспечение, эксплуатационная, техническая и пользовательская документация).
4. *Фаза внедрения*: окончательная версия системы, введённая в эксплуатацию.

Особенностью методологии *RUP* является то, что фазы не имеют жестких ограничений на вид выполняемых работ. Например, фаза уточнения включает в себя не только работы по проектированию системы, но и работы, связанные с программированием, тестированием. Также фаза конструирования не исключает

продолжения работ, связанных с построением бизнес-моделей и уточнением требований к системе. В связи с этим важное место в модели процесса *RUP* занимает понятие *дисциплины*.

Дисциплина RUP соответствует понятию технологического процесса и представляет собой последовательность действий, приводящую к получению значимого результата. В рамках *RUP* определены шесть основных дисциплин (технологических процессов) и три вспомогательных (поддерживающих).

Главным достоинством методологии *RUP* является удачное сочетание универсального и гибкого подхода к описанию процесса разработки (фазы, итерации, дисциплины) и формализованного подхода к формированию базы знаний (графические модели *UML*).

4.3. Гибкие методологии (*Agile*)

Гибкие методологии разработки программного обеспечения (*Agile software development*) – это группа методологий, которые ориентированы на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп. К классу гибких методологий разработки относятся, например, экстремальное программирование (*XP*), *Scrum*, *FDD* и др.

Большинство гибких методологий сводят процесс разработки к серии коротких циклов – итераций. Каждая итерация выглядит как программный проект в миниатюре и включает в себя все стандартные процессы (анализ, проектирование, программирование, тестирование и внедрение). Высокая частота выпуска и внедрения готовых версий позволяет разработчикам и заказчику своевременно обнаруживать проблемы постановки и реализации требований и исправлять их уже в следующей версии продукта.

Другим достоинством гибких методологий *Agile* является то, что они делают упор на непосредственное и частое общение всех задействованных в разработке лиц, включая заказчика или его полномочного представителя, определяющего требования к продукту. Это значительно ускоряет процесс выработки и согласования проектных решений. Также, отдавая предпочтение непо-

средственному общению, гибкие методологии уменьшают объём письменной документации по сравнению с другими методами.

В то же время характерные особенности гибких методологий на практике могут проявить себя как недостатки, связанные с планированием работ и управлением проектом.

Гибкий подход к выработке и реализации требований к системе не предполагает построения планов разработки на длительный срок. Это обусловлено тем, что заказчик в начале очередной итерации может сформулировать требования, противоречащие архитектуре текущей версии созданного и поставляемого продукта. В худшем случае это может привести к неоправданно большим затратам времени и средств на переработку архитектуры системы и повторную реализацию её компонентов.

Кроме того, считается, что организация процесса разработки с опорой на гибкие методологии мотивирует разработчиков решать все поступающие задачи наиболее простым и быстрым способом, часто не обращая внимания на качество решения задачи. Это относится как к потребительским качествам продукта, так и к качеству программного кода (форматирование кода, наличие комментариев, логическая непротиворечивость, следование принципам структурного и объектно-ориентированного программирования). Накопление и откладывание на неопределённый срок большого количества таких проблем может привести к тому, что в какой-то момент их совместное решение при выпуске очередной версии окажется крайне трудоёмким и дорогостоящим.

Таким образом, гибкие методологии *Agile* представляют собой оригинальный подход к организации процесса разработки ПО, успешность применения которого на практике сильно зависит от многих факторов и условий. Из приведённого выше описания достоинств и недостатков этого подхода видно, что гибкие методологии успешно применяются, если и разработчик, и заказчик технически и организационно компетентны, доверяют друг другу, ответственно и профессионально подходят к разработке программного продукта. В противном случае риски использования гибких методологий являются слишком высокими, и для организации процесса разработки следует выбрать другой, более формализованный подход.

5. ОСНОВЫ УПРАВЛЕНИЯ ПРОЕКТАМИ

5.1. Общие сведения о проектах

Информационная система разрабатывается как некоторый проект. Объектом разработки является программное обеспечение системы, поэтому проект разработки ИС часто называется *программным проектом*.

Многие особенности управления проектами и ключевые фазы разработки проекта (фазы ЖЦ объекта разработки) являются общими, не зависящими не только от предметной области, но и от характера проекта. Рассмотрим ряд вопросов управления проектами, которые являются общими для проектов любого вида.

Проект – это ограниченное по времени целенаправленное изменение отдельной системы с изначально четко определенными целями, достижение которых определяет завершение проекта, а также с установленными требованиями к срокам, результатам, рискам, рамкам расходования средств и ресурсов, а также к организационной структуре исполнителя проекта.

Можно выделить следующие основные отличительные признаки проекта как *объекта управления*:

1) изменчивость – целенаправленный перевод системы из существующего в некоторое желаемое состояние, описываемое в терминах целей проекта;
2) ограниченность конечной цели;
3) ограниченность продолжительности проекта;
4) ограниченность бюджета;
5) ограниченность требуемых ресурсов;
6) новизна для предприятия, на котором реализуется проект;
7) комплексность – наличие большого числа факторов, прямо или косвенно влияющих на прогресс и результаты проекта;
8) правовое и организационное обеспечение – создание специфической организационной структуры на время реализации проекта.

Рассматривая планирование проектов и управление ими, необходимо осознавать, что речь идет об управлении некоторым динамическим объектом. Поэтому система управления проектом должна быть достаточно гибкой, чтобы допускать возможность модификации планов без глобальных изменений в рабочей программе.

В системном плане проект может быть представлен «черным ящиком», входом которого являются технические требования и условия финансирования, а итогом работы – достижение требуемого результата. Выполнение работ обеспечивается наличием необходимых ресурсов:

1) материалов;

2) оборудования;

3) человеческих ресурсов.

Эффективность работ достигается за счет управления процессом реализации проекта, которое обеспечивает распределение ресурсов, координацию выполняемой последовательности работ и компенсацию внутренних и внешних возмущающих воздействий.

С точки зрения теории систем управления проект как объект управления должен быть наблюдаемым и управляемым. Для этого выделяются некоторые характеристики, по которым можно постоянно контролировать ход выполнения проекта (*свойство наблюдаемости*). Кроме того, необходимы механизмы своевременного воздействия на ход реализации проекта (*свойство управляемости*).

Свойство управляемости особенно актуально в условиях неопределенности и изменчивости предметной области, которые нередко сопутствуют проектам по разработке информационных систем. Для обоснования целесообразности и осуществимости проекта, анализа хода его реализации, а также для заключительной оценки степени достижения поставленных целей проекта и сравнения фактических результатов с запланированными существует ряд характеристик проекта. К важнейшим из них относятся технико-экономические показатели:

1) объем работ;

2) сроки выполнения проекта;

3) себестоимость проекта;
4) экономическая эффективность, обеспечиваемая реализацией проекта;
5) социальная и общественная значимость проекта.

Проекты могут сильно отличаться по сфере приложения, составу, предметной области, масштабам, длительности, составу участников, степени сложности, значимости результатов и т.п.

Класс проекта определяется по его составу и структуре:

1) *монопроект* – это отдельный проект, который может быть любого типа, вида и масштаба;
2) *мультипроект* – это комплексный проект, состоящий из ряда монопроектов и требующий применения многопроектного управления.

Тип проекта определяется по основным сферам деятельности, в которых он осуществляется:

1) технический;
2) организационный;
3) экономический;
4) социальный;
5) смешанный.

Разработка информационных систем относится к техническим проектам, которые имеют следующие особенности:

1. Главная цель проекта четко определена, но отдельные цели должны уточняться по мере достижения частных результатов.
2. Срок завершения и продолжительность проекта определены заранее. Желательно их точное соблюдение, однако они могут корректироваться в зависимости от полученных промежуточных результатов и общего прогресса проекта.

Масштаб проекта определяется по размерам бюджета и количеству участников: мелкие, малые проекты, проекты, крупные проекты. Можно также рассматривать масштабы проектов в более конкретной форме – отраслевые, корпоративные, ведомственные проекты, проекты одного предприятия.

5.2. Организация процесса разработки программного обеспечения

Процесс разработки программного обеспечения – это совокупность различных, связанных друг с другом видов деятельности, методов, методик и шагов, используемых для разработки и эволюции ПО и связанных с ним продуктов (проектных планов, документации, программного кода, тестов, пользовательской документации и т.д.).

Процесс разработки ПО невозможно стандартизировать или систематизировать таким образом, чтобы любая группа разработчиков могла использовать его автоматически. Каждая организация должна разработать свою собственную модель процесса или приспособить некоторый настраиваемый шаблон процесса под свои нужды.

Не существует универсального процесса разработки в виде набора методик, правил и предписаний, одинаково хорошо подходящих для любых программных продуктов, любых заказчиков, любого состава команды разработчиков. Каждый новый процесс разработки, осуществляемый некоторой командой в рамках определенного проекта, имеет большое количество особенностей.

Процесс разработки ПО может в точности следовать положениям одной из основных моделей ЖЦ, использовать вариант одной из моделей или опираться на несколько моделей одновременно, преследуя цель использовать достоинства и компенсировать недостатки каждой из моделей. Для построения эффективного процесса разработки ИС необходимо иметь представление о свойствах, характеристиках, достоинствах и недостатках различных моделей ЖЦ.

Разработка информационной системы является весьма сложной задачей, процесс решения которой разбивается на определённое количество этапов. Одним из ключевых этапов создания ИС является её проектирование на основе заданной спецификации.

Спецификация ИС – это набор исходных требований и параметров, которым должна удовлетворять информационная система в результате её создания.

Проект ИС – это проектно-конструкторская и технологическая документация, в которой представлено описание проектных решений по созданию и эксплуатации информационной системы.

Проектирование ИС – процесс преобразования спецификации (входной информации) в проект информационной системы (результат проектирования).

К числу ключевых этапов создания ИС также относятся: анализ требований, реализация (программирование), тестирование и внедрение. Объединение этих этапов в один процесс приводит к понятию жизненного цикла ИС.

Жизненный цикл (ЖЦ) информационной системы – непрерывный процесс, который начинается с момента принятия решения о необходимости создания системы и заканчивается в момент её полного изъятия из эксплуатации.

Согласно стандарту ГОСТ Р ИСО/МЭК 12207–2010 жизненный цикл ИС основывается на трёх видах процессов:

1) основные (разработка, эксплуатация и др.);
2) вспомогательные (верификация, оценка, управление конфигурацией, документирование и др.);
3) организационные (создание инфраструктуры и др.).

С другой стороны, в жизненном цикле ИС выделяют ряд ключевых этапов. Согласно стандарту ГОСТ Р ИСО/МЭК 15288–2005 рассматривают следующие этапы жизненного цикла:

1. *Замысел*. – Замысел новой системы. Оценка реализуемости, предварительное планирование.
2. *Разработка*. – Проект системы, или прототип системы, или конечный программный продукт.
3. *Производство*. – Выпуск продукции. Предоставление продукции пользователям.
4. *Эксплуатация*. – Обеспечение декларированных характеристик системы в процессе её эксплуатации.
5. *Сопровождение*. – Техническое обслуживание и сопровождение, обеспечивающее непрерывное функционирование системы.
6. *Снятие с эксплуатации*. – Прекращение использования системы.

6. ПРОГРАММНЫЕ СРЕДСТВА ПОДДЕРЖКИ ЖИЗНЕННОГО ЦИКЛА

6.1. *CASE*-технологии и *CASE*-средства

Первым шагом в проектировании ИС является получение формального описания предметной области, построение полных и непротиворечивых функциональных и информационных моделей системы. Это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Следует также учитывать, что в процессе создания и функционирования ИС потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких систем. Указанные сложности способствовали появлению специальных технологий (*CASE*-технологий) и программных средств (*CASE*-средств), призванных повысить эффективность разработки ПО и создания ИС.

Термин *CASE* (*Computer Aided Software/System Engineering*) используется в настоящее время в весьма широком смысле. Первоначальное значение термина *CASE*, ограниченное вопросами автоматизации разработки только программного обеспечения, в настоящее время обрело новый смысл, охватывающий процесс разработки сложных информационных систем в целом. Широко используются два тесно связанных друг с другом понятия: *CASE*-технология и *CASE*-средства.

CASE-технология представляет собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных информационных систем, которая поддерживается комплексом взаимосвязанных программных средств автоматизации.

CASE-средства – это программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО и баз данных, генерацию кода, тестирование, документирование,

обеспечение качества, конфигурационное управление, управление проектом и т.д.

Современные *CASE*-средства охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО. Наиболее трудоемкими этапами создания ИС являются этапы анализа требований к системе и её проектирования, в процессе которых *CASE*-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

6.2. Возможности и особенности *CASE*-средств

В наиболее полном виде *CASE*-средства и поддерживаемые ими *CASE*-технологии обладают следующими ключевыми возможностями:

1. *Единая база данных проекта*. Основой *CASE*-технологии является использование единой базы данных проекта (*репозитория*) для хранения всей информации, которая может совместно использоваться разработчиками в процессе создания системы. Репозиторий может хранить объекты различных типов: структурные диаграммы, эскизы экранных форм, модели данных, описание алгоритмов обработки данных и т.д.

2. *Единый графический язык*. *CASE*-средства обеспечивают всех участников проекта, включая заказчика, единым строгим, наглядным и интуитивно понятным графическим языком, позволяющим получать обозримые компоненты с простой и ясной структурой.

3. *Интеграция средств*. На основе репозитория осуществляются интеграция *CASE*-средств и совместное использование системной информации всеми разработчиками.

4. *Поддержка коллективной разработки и управления проектом*. *CASE*-технология поддерживает групповую работу над

проектом, планирование, контроль, руководство и взаимодействие, то есть функции, необходимые в процессе коллективной разработки и сопровождения проектов.

5. *Макетирование*. *CASE*-технология дает возможность быстро строить макеты (прототипы) будущей ИС, что позволяет заказчику уже на ранних этапах разработки системы оценить, соответствует ли система его требованиям.

6. *Генерация документации*. Вся документация по проекту генерируется автоматически на базе репозитория. Документация всегда является актуальной, т.к. любые изменения в проекте автоматически отражаются в объектах репозитория.

7. *Верификация проекта*. *CASE*-технология обеспечивает автоматическую верификацию и контроль проекта на полноту и состоятельность на ранних этапах разработки, что влияет на успех разработки в целом.

8. *Автоматическая генерация программного кода*. Генерация программного кода осуществляется на основе объектов репозитория и позволяет автоматически построить значительную часть исходного кода программ на языках высокого уровня.

9. *Сопровождение*. Сопровождение программной системы в рамках *CASE*-технологии характеризуется сопровождением проекта, а не программных кодов.

Далеко не все *CASE*-средства поддерживают все указанные выше возможности. Поэтому обычно к *CASE*-средствам относят любой программный продукт, автоматизирующий ту или иную совокупность процессов жизненного цикла ПО и обладающий следующими основными характерными особенностями:

1) наличие мощных графических средств для описания и документирования ИС;
2) интеграция отдельных компонентов *CASE*-средств, обеспечивающая управляемость процесса разработки ИС;
3) использование специальным образом организованного хранилища проектных метаданных (репозитория).

ЧАСТЬ II. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Описание лабораторного практикума включает в себя учебно-методические материалы к выполнению шести лабораторных работ по всем темам рабочей программы дисциплины «Управление жизненным циклом информационных систем».

Работа выполняется как во время аудиторных занятий, так и в виде самостоятельной внеаудиторной работы. Выполнение каждой лабораторной работы состоит из трёх этапов:

1. Подготовка и получение допуска к работе.
2. Получение индивидуального задания и выполнение основной части работы.
3. Оформление и защита отчёта о проделанной работе.

В начале каждой лабораторной работы выполняется повторение теоретического материала и проверка готовности к выполнению работы с помощью контрольных вопросов. После получения допуска к выполнению работы выдаётся индивидуальный вариант задания для самостоятельной работы. На заключительном этапе оформляется отчёт о проделанной работе с описанием полученных результатов и выполняется процедура защиты отчёта.

Процедура защиты отчёта заключается в проверке:

1) правильности структуры и оформления отчёта;
2) корректности полученных результатов;
3) способности дать объяснение и необходимое обоснование полученным результатам.

Отчет должен включать в себя:

1. Титульный лист.
2. Задание на лабораторную работу.
3. Содержание отчёта.
4. Описание результатов по каждой части задания.
5. Приложение (диаграммы *UML*, тексты программ, содержание проектных документов и т.д.).

Лабораторная работа № 1
ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Цели и задачи лабораторной работы

Целями выполнения лабораторной работы являются:

1. Закрепление знаний о видах и назначении информационных систем (ИС). Изучение области применения и функциональных возможностей современных ИС.
2. Приобретение практических навыков поиска, обработки и анализа информации по заданной теме в сети интернет.
3. Приобретение навыков составления документа-обоснования для внедрения информационной системы.

В процессе выполнения лабораторной работы решаются следующие задачи:

1. Выполняется поиск и анализ информации: о заданном виде ИС, о конкретных информационных системах заданного вида.
2. Разрабатывается пример возможного применения одной из информационных систем заданного вида в деятельности некоторого объекта автоматизации (предприятия, организации).
3. Составляется документ-обоснование на внедрение информационной системы.

Краткие теоретические сведения

Информационная система – это совокупность программного обеспечения и электронного информационного хранилища (базы данных), разрабатываемая как единая система и предназначенная для автоматизации определённого рода деятельности.

По роли, которую информационные системы играют в профессиональной деятельности, и решаемым ими задачам можно выделить следующие виды систем:

1) системы управления;

2) вычислительные информационные системы;
3) поисково-справочные информационные системы;
4) системы поддержки принятия решений;
5) информационные обучающие системы.

В зависимости от степени автоматизации выделяют ручные, автоматизированные и автоматические ИС.

Контрольные вопросы для допуска к работе

1. Автоматизация бизнес-процессов.
2. Информационные системы.
3. Виды информационных систем, их назначение и состав.
4. Технологии разработки информационных систем.
5. Методологии разработки программного обеспечения.
6. Процесс разработки программного обеспечения.
7. Управление разработкой программного обеспечения.
8. Проектирование информационных систем.
9. Этапы проектирования.
10. Задачи и результаты проектирования.

Порядок выполнения работы

Вариант индивидуального задания определяет один из видов современных информационных систем.

В процессе выполнения лабораторной работы необходимо:

1. Найти информацию, характеризующую назначение и область применения заданного вида информационных систем.
2. Определить, к какому классу относится заданный вид информационных систем (по характеру использования информации, по сфере применения, по способу организации, по уровню и масштабу решаемых задач).
3. Составить общее описание заданного вида информационных систем.
4. Найти описание нескольких (не менее двух) современных информационных систем, относящихся к заданному виду.
5. Сформулировать краткое описание назначения и функциональных возможностей каждой из информационных систем по отдельности. Указать на характеристики и

свойства, которые являются общими для всех рассматриваемых ИС.

6. Составить таблицу отличий между информационными системами. Указать на их индивидуальные особенности, различающиеся количественные и качественные характеристики.
7. Разработать пример возможного применения одной из информационных систем в деятельности некоторого объекта автоматизации (предприятия или организации). Вид деятельности объекта автоматизации выбирается самостоятельно.
8. Составить документ-обоснование для внедрения информационной системы. Описать, чего позволит достичь внедрение информационной системы с точки зрения повышения эффективности работы объекта автоматизации (организации, предприятия).

Варианты индивидуальных заданий

1. Корпоративные информационные системы (КИС).
2. Системы автоматизации бизнес-процессов (САБП).
3. Геоинформационные системы (ГИС).
4. Системы электронного документооборота (СЭДО).
5. Системы управления корпоративным контентом.
6. Системы планирования ресурсов предприятия.
7. Системы управления взаимоотношениями с клиентами.
8. Системы управления веб-контентом.
9. Интеллектуальные информационные системы.
10. Системы поддержки принятия решений.
11. Информационно-управляющие системы.
12. Информационно-вычислительные системы.
13. Информационно-справочные системы.
14. Обучающие системы.
15. Поисковые системы.
16. Системы автоматизированного проектирования (САПР).

Лабораторная работа № 2
МОДЕЛИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Цели и задачи лабораторной работы

Целями выполнения лабораторной работы являются:

1. Закрепление имеющихся знаний о технологиях и методологиях моделирования информационных систем.
2. Приобретение навыков объектно-ориентированного анализа, моделирования и проектирования ИС.
3. Приобретение навыков разработки моделей ИС в виде диаграмм, построенных с применением унифицированного языка моделирования *UML*.

В процессе выполнения лабораторной работы решаются следующие задачи:

1. Выполняется разработка концептуальных моделей ИС для описания автоматизируемых бизнес-процессов с помощью диаграмм деятельности и диаграмм последовательности.
2. Выполняется разработка логических моделей ИС для описания требований к системе с помощью диаграмм прецедентов и диаграмм классов.
3. Выполняется разработка физических моделей ИС для описания конкретного способа реализации системы с помощью диаграмм базы данных, диаграмм компонентов и диаграмм развёртывания.

Краткие теоретические сведения

Унифицированный язык моделирования UML – это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех компонентов, создаваемых при разработке программных систем.

Язык *UML* является объектно-ориентированным языком. Его использование основывается на понимании общих принципов *объектно-ориентированного анализа и проектирования*:

1. *Принцип абстрагирования* предписывает включать в модель только те аспекты проектируемой системы, которые имеют непосредственное отношение к выполнению системой своих функций.
2. *Принцип многомодельности* означает, что никакое единственное представление системы не является достаточным для адекватного выражения всех ее особенностей.
3. *Принцип иерархического построения моделей сложных систем* предписывает рассматривать процесс построения моделей на разных уровнях абстрагирования или детализации в рамках фиксированных представлений.

Диаграмма UML – это графическое представление набора элементов, изображаемое в виде связанного графа с вершинами (сущностями) и ребрами (отношениями), используемое для визуализации системы с разных точек зрения.

Диаграммы *UML* используются для описания различных аспектов функционирования и структуры ИС на разных стадиях создания системы и, соответственно, на разных этапах моделирования: концептуального, логического и физического.

Контрольные вопросы для допуска к работе

1. Моделирование информационных систем.
2. Виды моделей информационных систем.
3. Объектно-ориентированный анализ и проектирование.
4. Технологии, языки и средства моделирования.
5. Язык унифицированного моделирования *UML*.
6. Диаграммы языка *UML*: структурные диаграммы, диаграммы поведения, диаграммы взаимодействия.
7. Инструментальные средства моделирования ИС.
8. Применение *UML* при проектировании ИС.

Порядок выполнения работы

Вариант индивидуального задания определяет ИС, для создания которой необходимо разработать совокупность моделей

системы в виде комплекта диаграмм *UML*. Построенные модели ИС должны описывать различные аспекты проектирования и разработки системы на разных стадиях её жизненного цикла.

В процессе выполнения лабораторной работы необходимо:

1. Разработать модель прецедентов, описывающую бизнес-процессы организации с точки зрения внешнего пользователя (клиента) и отражающую взгляд на деятельность организации извне. Результатом моделирования являются диаграммы деятельности и диаграммы прецедентов.
2. Разработать модель бизнес-объектов, описывающую выполнение бизнес-процессов организации ее внутренними исполнителями. Основными компонентами модели являются внешние и внутренние исполнители. Результатом моделирования являются диаграммы последовательности.
3. Разработать концептуальную модель данных, описывающую объекты предметной области и связи между ними. Результатом моделирования являются диаграммы классов и диаграммы объектов.
4. Разработать описание требований к системе. Результатом является исчерпывающий перечень функций, которые должны быть реализованы в системе, и подробное описание необходимой реализации этих функций.
5. Разработка моделей базы данных и приложений, представляющих собой детальное описание проекта базы данных и клиентских приложений ИС. Результатом моделирования являются диаграммы компонентов и диаграммы базы данных.
6. Разработать проект физической реализации информационной системы. Результатом проектирования являются диаграммы развёртывания и диаграммы компонентов.

Варианты индивидуальных заданий

1. Моделирование предметной области:

1. Телефонный справочник.
2. Библиотека.
3. Издательство.
4. Поликлиника.

5. Школа.
6. Ателье по пошиву и ремонту одежды.
7. Оптовый склад.
8. Торгово-закупочное предприятие.
9. Автосалон.
10. Продажа подержанных автомобилей.
11. Автосервис.
12. Пассажирское автопредприятие.
13. Диспетчерская служба такси.
14. Агентство по продаже авиабилетов.
15. Туристическое агентство.
16. Гостиница.

2. Моделирование информационной системы:

1. ИС «Телефонный справочник» (поисковая система).
2. ИС «Библиотека» (информационно-справочная система, поисковая система).
3. ИС «Издательство» (СЭДО, САБП).
4. ИС «Поликлиника» (СЭДО, информационно-справочная система).
5. ИС «Школа» (обучающая система, информационно-справочная система).
6. ИС «Ателье» (САБП).
7. ИС «Склад» (САБП).
8. ИС «Торговля» (САБП, СЭДО).
9. ИС «Автосалон» (САБП, СЭДО).
10. ИС «Продажа подержанных автомобилей» (информационно-справочная система, поисковая система).
11. ИС «Автосервис» (САБП).
12. ИС «Пассажирское автопредприятие» (САБП, СЭДО).
13. ИС «Диспетчерская служба такси» (ГИС, СЭДО).
14. ИС «Агентство по продаже авиабилетов» (информационно-справочная система, поисковая система).
15. ИС «Туристическое агентство» (информационно-справочная система, поисковая система).
16. ИС «Гостиница» (информационно-справочная система, СЭДО).

Лабораторная работа № 3
ЖИЗНЕННЫЙ ЦИКЛ ИНФОРМАЦИОННЫХ СИСТЕМ

Цели и задачи лабораторной работы

Целями выполнения лабораторной работы являются:

1. Закрепление имеющихся знаний о моделях жизненного цикла ИС и способах их применения для разработки программного обеспечения.
2. Приобретение навыков анализа требований, условий и ограничений проекта создания ИС и оценки трудоёмкости его реализации.
3. Приобретение навыков составления планов разработки ИС на основе разных моделей жизненного цикла.

В процессе выполнения лабораторной работы решаются следующие задачи:

1. Выполняется анализ постановки задачи. Готовятся исходные данные для планирования. Формулируются ограничения и условия разработки.
2. Разрабатываются прототипы документов: «Техническое задание», «Технический проект», «План тестирования», «План ввода в эксплуатацию».
3. Составляется календарный план разработки ИС.

Краткие теоретические сведения

Жизненный цикл (ЖЦ) информационной системы – непрерывный процесс, который начинается с момента принятия решения о необходимости создания системы и заканчивается в момент её полного изъятия из эксплуатации.

Основными этапами ЖЦ ИС являются:

1. Анализ (разработка требований).
2. Проектирование (создание проекта).
3. Реализация (программирование).

4. Тестирование (исправление ошибок).
5. Внедрение (ввод в эксплуатацию).

Модель жизненного цикла ИС – структура, описывающая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного обеспечения в течение всей жизни ИС, от определения требований до завершения её использования.

К настоящему времени наибольшее распространение получили следующие основные модели ЖЦ:

1) каскадная (водопадная) модель и её варианты;
2) инкрементная модель;
3) спиральная модель.

Каскадная или *водопадная* модель ЖЦ является классической моделью однократного прохода, которая описывает линейную последовательность этапов создания ИС.

Каскадная модель с промежуточным контролем является модификацией каскадной модели ЖЦ, которая по окончании текущего этапа предусматривает возможность возврата на предыдущий этап для уточнения требований.

V-образная каскадная модель ЖЦ является развитием классической модели. Её отличает то, что каждому шагу этапов анализа, проектирования и реализации соответствует отдельный шаг на этапах тестирования и внедрения.

Инкрементная модель ЖЦ отличается от классической каскадной тем, что в ней существует сразу несколько комплектов требований к системе (спецификаций) с разной степенью полноты. Вся разработка делится на заданное количество шагов (итераций, инкрементов).

Спиральная модель ЖЦ относится к эволюционным моделям. Каждый виток раскручивающейся спирали соответствует разработке одной (начальной, промежуточной или окончательной) версии ИС и представляет собой полный цикл разработки, начиная с анализа и заканчивая внедрением.

Прототип – версия ИС, предназначенная для демонстрации заказчику некоторых ключевых свойств будущего продукта. Создание прототипа позволяет вовлечь заказчика в разработку информационной системы в самом начале работы.

Контрольные вопросы для допуска к работе

1. Современные методологии разработки информационных систем.
2. Жизненный цикл информационных систем.
3. Этапы жизненного цикла: анализ, проектирование, программирование, тестирование, эксплуатация.
4. Стандартные модели жизненного цикла.
5. Каскадная модель жизненного цикла.
6. Преимущества и недостатки каскадной модели жизненного цикла.
7. Каскадная модель с промежуточным контролем.
8. V-образная каскадная модель.
9. Итеративная модель жизненного цикла.
10. Спиральная модель жизненного цикла.

Порядок выполнения работы

Вариант индивидуального задания определяет информационную систему, для создания которой необходимо составить план разработки на основе каскадной и спиральной моделей жизненного цикла.

В процессе выполнения лабораторной работы необходимо:

1. Подготовить исходные данные. Исходными данными для планирования являются:
   1.1. Общее описание некоторой ИС (назначение, область применения, решаемые задачи, технологические особенности реализации и внедрения).
   1.2. Ограничения и условия разработки (требования заказчика, возможности команды разработчиков, сроки разработки, бюджет проекта и т.д.).
2. Составить план разработки ИС с применением каскадного подхода:
   2.1. Составить эскизный план разработки ИС на основе каскадной модели ЖЦ.
   2.2. Для этапа «Анализ требований» составить документ «Техническое задание» с подробным описанием функциональных требований к ИС.

2.3. Для этапа «Проектирование» составить документ «Технический проект» с описанием проектных решений (архитектура системы, логическая структура базы данных, решения по реализации пользовательского интерфейса и т.д.).
2.4. Для этапа «Тестирование» составить документ «План тестирования» с описанием методики тестирования и контрольных тестов.
2.5. Для этапа «Внедрение» составить документ «План ввода ИС в эксплуатацию».
2.6. Уточнить параметры календарного плана разработки ИС, учитывая ограничения и условия разработки.
2.7. Объединить календарный план разработки и составленные документы в единый отчёт «Разработка ИС на основе каскадной модели ЖЦ».

3. Составить план разработки ИС с применением итеративного подхода:
3.1. Разделить весь процесс создания и внедрения ИС на несколько итераций.
3.2. На основе имеющихся документов (см. пункты 2.2 – 2.5) для каждой итерации составить отдельный комплект документов.
3.3. Составить календарный план итеративной разработки ИС.
3.4. Объединить план итеративной разработки и составленные документы в единый отчёт «Разработка ИС на основе спиральной модели ЖЦ».

Варианты индивидуальных заданий

В качестве списка вариантов индивидуальных заданий используется перечень информационных систем из лабораторной работы № 2.

Лабораторная работа № 4
СОВРЕМЕННЫЕ МЕТОДОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Цели и задачи лабораторной работы

Целями выполнения лабораторной работы являются:

1. Закрепление имеющихся знаний о современных методологиях разработки программного обеспечения.
2. Приобретение навыков анализа требований, условий и ограничений проекта создания ИС и оценки трудоёмкости его реализации.
3. Приобретение навыков составления планов разработки ИС на основе положений и рекомендаций различных методологий разработки ПО.

В процессе выполнения лабораторной работы решаются следующие задачи:

1. Выполняется анализ постановки задачи. Готовятся исходные данные для планирования. Формулируются ограничения и условия разработки ИС.
2. Разрабатывается документ «Техническое задание», описывающий требования к ИС.
3. Составляется план итеративной разработки ИС на основе положений и рекомендаций методологии *MSF*.
4. Составляется план итеративной разработки ИС на основе положений и рекомендаций методологии *RUP*.

Краткие теоретические сведения

Методология *Microsoft Solutions Framework* (разработана компанией *Microsoft*) описывает подходы и принципы управления людьми и рабочими процессами для организации процесса разработки программного обеспечения.

Руководство по *MSF* структурно состоит из пяти документов, каждый из которых описывает определённую *модель* или *дисциплину*: модель процессов, модель проектной группы, дисциплину управления проектами, дисциплину управления рисками, дисциплину управления подготовкой.

Элементы каскадной модели ЖЦ реализуются в модели процессов *MSF* в виде системы *вех* и *фаз*. *Вехи* – это контрольные точки проекта, характеризующие достижение в его рамках какого-либо существенного (промежуточного или конечного) результата. *Фазы* – это этапы (стадии) между вехами.

Методология *Rational Unified Process* (разработана компанией *Rational Software*) описывает, как эффективно применять коммерчески обоснованные и практически опробованные подходы к разработке программных продуктов.

С точки зрения организации процесса разработки методология *RUP* использует итеративную модель ЖЦ. Процесс разработки состоит из четырёх фаз, каждая из которых включает в себя одну или несколько итераций.

Дисциплина RUP соответствует понятию технологического процесса и представляет собой последовательность действий, приводящую к получению значимого результата. В рамках *RUP* определены шесть основных дисциплин (технологических процессов) и три вспомогательных (поддерживающих).

Контрольные вопросы для допуска к работе

1. Методология *MSF*. Модели и дисциплины *MSF*.
2. Модель процесса *MSF*. Итеративная разработка.
3. Структура модели жизненного цикла *MSF*. Вехи и фазы.
4. Методология *RUP*.
5. Модель процесса разработки *RUP*. Фазы и итерации.
6. Дисциплины *RUP*.

Порядок выполнения работы

Вариант индивидуального задания определяет ИС, для создания которой необходимо составить план разработки на основе положений и рекомендаций двух методологий разработки программного обеспечения: *MSF* и *RUP*.

В процессе выполнения лабораторной работы необходимо:

1. Подготовить исходные данные для планирования, взяв за основу результаты, полученные при выполнении лабораторной работы № 3:
   1.1. Общее описание некоторой ИС.
   1.2. Ограничения и условия разработки.
2. Составить документ «Техническое задание» с подробным описанием концептуальных и функциональных требований к ИС.
3. Составить план разработки ИС с применением положений и рекомендаций методологии *Microsoft Solutions Framework*:
   3.1. Составить эскизный план разработки ИС на основе модели ЖЦ, описанной в модели процессов *MSF*.
   3.2. Определить примерное количество итераций, необходимое для разработки ИС.
   3.3. Рассматривая последовательно каждую итерацию, сформировать комплект проектной документации, состоящий из документов «План итерации № …» План каждой итерации должен включать в себя следующие разделы:
      3.3.1. для фазы «Выработка концепции» – постановку задачи на разработку соответствующей версии ИС;
      3.3.2. для фазы «Планирование» – описание организационных и технических проектных решений по разработке ИС;
      3.3.3. для фазы «Разработка» – характеристику ожидаемых результатов разработки очередной версии ИС;
      3.3.4. для фазы «Стабилизация» – набор контрольных тестов для валидации и верификации программного обеспечения ИС;
      3.3.5. для фазы «Внедрение» – описание мероприятий по переходу пользователей на новую версию ИС.

3.4. Объединить документы, составленные по отдельным итерациям, в единый отчёт «Планирование разработки ИС на основе методологии *MSF*».

4. Составить план разработки ИС с применением положений и рекомендаций методологии *Rational Unified Process*:
   4.1. Составить эскизный план разработки ИС на основе модели ЖЦ, описанной в модели процессов *RUP*.
   4.2. Определить примерное количество итераций, необходимое для разработки ИС. Распределить итерации по фазам процесса разработки (начальная фаза, фаза уточнения, фаза конструирования, фаза внедрения).
   4.3. Рассматривая последовательно каждую фазу, сформировать комплект проектной документации, состоящий из документов «План фазы …» План каждой фазы должен включать в себя следующие разделы:
      4.3.1. постановку задачи на разработку соответствующей версии ИС;
      4.3.2. описание организационных и технических проектных решений по разработке ИС;
      4.3.3. характеристику ожидаемых результатов разработки очередной версии ИС;
      4.3.4. набор контрольных тестов для валидации и верификации программного обеспечения ИС;
      4.3.5. описание мероприятий по переходу пользователей на новую версию ИС.
   4.4. Объединить документы, составленные по отдельным фазам процесса разработки, в единый отчёт «Планирование разработки ИС на основе методологии *RUP*».

Варианты индивидуальных заданий

В качестве списка вариантов индивидуальных заданий используется перечень информационных систем из лабораторной работы № 2.

Лабораторная работа № 5
ОСНОВЫ УПРАВЛЕНИЯ ПРОЕКТАМИ

Цели и задачи лабораторной работы

Целями выполнения лабораторной работы являются:

1. Закрепление имеющихся знаний о проектах разработки ПО, методах управления программными проектами, стандартах процесса разработки и жизненного цикла ПО.
2. Приобретение навыков оценки стоимости программного проекта на основе имеющейся информации о требованиях к ПО и трудоёмкости разработки.
3. Приобретение навыков планирования и организации процесса разработки ПО с учётом различных условий и ограничений.

В процессе выполнения лабораторной работы решаются следующие задачи:

1. На основе требований к ИС определяются характеристики программного проекта. Оценивается сложность, масштаб и реализуемость проекта.
2. Формулируются задачи, выполнение которых необходимо для реализации программного проекта. Определяется трудоёмкость выполнения отдельных задач. Оценивается общая стоимость реализации проекта.
3. Составляются календарные планы разработки программного продукта с учётом конкретных условий разработки.

Краткие теоретические сведения

Проект – это ограниченное по времени целенаправленное изменение отдельной системы с изначально четко определенными целями, достижение которых определяет завершение проекта, а также с установленными требованиями к срокам, результатам, рискам, рамкам расходования средств и ресурсов, а также к организационной структуре исполнителя проекта.

Процесс разработки программного обеспечения – это совокупность различных, связанных друг с другом видов деятельности, методов, методик и шагов, используемых для разработки и эволюции ПО и связанных с ним продуктов (проектных планов, документации, программного кода, тестов, пользовательской документации и т.д.).

Спецификация ИС – это набор исходных требований и параметров, которым должна удовлетворять информационная система в результате её создания.

Проект ИС – это проектно-конструкторская и технологическая документация, в которой представлено описание проектных решений по созданию и эксплуатации информационной системы.

Проектирование ИС – процесс преобразования спецификации (входной информации) в проект информационной системы (результат проектирования).

Стандарт ГОСТ Р ИСО/МЭК 12207–2010 определяет три вида процессов ЖЦ: основные (разработка, эксплуатация и др.), вспомогательные (верификация, документирование и др.) и организационные (создание инфраструктуры и др.)

Стандарт ГОСТ Р ИСО/МЭК 15288–2005 определяет следующие этапы жизненного цикла: замысел, разработка, производство, эксплуатация, сопровождение, снятие с эксплуатации.

Контрольные вопросы для допуска к работе

1. Проект. Управление проектами.
2. Признаки проекта как объекта управления.
3. Характеристики проекта: класс, тип, масштаб, сложность, реализуемость.
4. Программный проект. Особенности управления программными проектами.
5. Методы оценки стоимости программного проекта.
6. Процесс разработки программного обеспечения.
7. Спецификация информационной системы.
8. Проектирование системы. Проект системы.
9. Стандарты ГОСТ этапов и процессов ЖЦ ИС.

Порядок выполнения работы

Вариант индивидуального задания определяет информационную систему, создание которой рассматривается как программный проект, требующий соответствующих решений, документов и действий для планирования и организации процесса разработки программного обеспечения.

В процессе выполнения лабораторной работы необходимо:

1. Поставить задачу создания ИС как проект разработки соответствующего программного обеспечения. Охарактеризовать проект с точки зрения целей, задач и результатов работы.
2. Выполнить анализ функциональных требований к ИС. Оценить сложность, масштаб и реализуемость проекта, учитывая требования к срокам реализации проекта, бюджет проекта, организационную структуру исполнителя проекта.
3. От описания функциональных требований к ИС перейти к перечню задач, выполнение которых необходимо для реализации программного проекта. Систематизировать и детализировать задачи. Выполнить декомпозицию сложных задач (разбить сложную задачу на отдельные подзадачи).
4. Определить трудоёмкость выполнения отдельных типовых задач. Вычислить общую трудоёмкость решения всех задач. Оценить стоимость реализации всего программного проекта.
5. Составить календарные планы разработки ИС с учётом конкретных условий разработки: численности и квалификации персонала, используемой модели ЖЦ и методологии разработки ПО, сроков реализации проекта и др.
6. Оформить план реализации проекта в виде документа, охватывающего все этапы жизненного цикла ИС.

Варианты индивидуальных заданий

В качестве списка вариантов индивидуальных заданий используется перечень информационных систем из лабораторной работы № 2.

Лабораторная работа № 6
ПРОГРАММНЫЕ СРЕДСТВА ПОДДЕРЖКИ ЖИЗНЕННОГО ЦИКЛА

Цели и задачи лабораторной работы

Целями выполнения лабораторной работы являются:

1. Закрепление имеющихся знаний о *CASE*-технологиях, применяемых для автоматизации процесса разработки информационных систем.
2. Приобретение навыков выбора средств автоматизации процесса разработки ИС (*CASE*-средств) с учётом принятой модели жизненного цикла и используемой методологии разработки программного обеспечения.
3. Приобретение навыков применения *CASE*-технологии и *CASE*-средств для решения задач, возникающих в процессе создания информационных систем.

В процессе выполнения лабораторной работы решаются следующие задачи:

1. Формулируются требования к функциональным возможностям *CASE*-средств, выбираемым для автоматизации процесса разработки заданной ИС.
2. Описывается реализация и порядок использования наиболее существенных компонентов *CASE*-технологии: репозитория, средств графического моделирования, технологий взаимодействия между разработчиками, средств макетирования, прототипирования и автоматической генерации программного кода.
3. Разрабатывается документ, описывающий порядок применения *CASE*-технологии и *CASE*-средств для автоматизации процесса разработки заданной ИС.

Краткие теоретические сведения

CASE-технология представляет собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных программных систем, которая поддерживается комплексом взаимосвязанных программных средств автоматизации.

Основой *CASE*-технологии является использование единой базы данных (*репозитория*) для хранения всей информации, которая может использоваться в процессе создания системы. Репозиторий может хранить объекты различных типов: структурные диаграммы, эскизы экранных форм, модели данных, описание алгоритмов обработки данных и т.д.

CASE-средства – это программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулирование требований, проектирование прикладного ПО и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление, управление проектом и т.д.

К *CASE*-средствам относят любой программный продукт, обладающий следующими основными характерными особенностями:

1) наличие мощных графических средств для описания и документирования ИС;
2) интеграция отдельных компонентов *CASE*-средств, обеспечивающая управляемость процесса разработки ИС;
3) использование специальным образом организованного хранилища проектных метаданных (репозитория).

Быстрая разработка приложений *RAD* (*Rapid Application Development*) является одной из современных методологий разработки ПО. Методологию *RAD* связывают с технологией *визуального программирования* и применением современных *интегрированных сред разработки* (ИСР) программного обеспечения.

Одним из основных принципов методологии *RAD* является необходимость применения *CASE*-средств, обеспечивающих целостность проекта на всех этапах его реализации. Все модели и прототипы должны быть получены с применением тех *CASE*-средств, которые будут использоваться в дальнейшем при построении системы. Данное требование вызвано тем, что при пе-

редаче информации о проекте с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет избежать этой опасности. Также *CASE*-средства применяются для автоматической генерации программного кода при помощи автоматических генераторов, получающих информацию непосредственно из репозитория.

Контрольные вопросы для допуска к работе

1. Автоматизация процессов разработки ИС.
2. Средства автоматизации разработки программного обеспечения.
3. CASE-технология: назначение, состав и ключевые возможности.
4. CASE-средства: назначение и выполняемые функции.
5. Репозиторий. Роль репозитория в автоматизации процессов разработки ИС.
6. Подходы к автоматизации процессов разработки ИС.
7. Структурный подход (информационные, функциональные, структурные модели).
8. Объектно-ориентированный подход.
9. Методология быстрой разработки приложений RAD.
10. Интегрированные среды разработки ПО.

Порядок выполнения работы

Вариант индивидуального задания определяет информационную систему, процесс разработки которой необходимо автоматизировать с применением *CASE*-технологии и соответствующих программных средств.

В процессе выполнения лабораторной работы необходимо:

1. Сформулировать требования к *CASE*-технологии и функциональным возможностям *CASE*-средств, выбираемым для автоматизации процесса разработки ИС.
2. Описать структуру и содержание репозитория, используемого в качестве единой базы данных проекта. Указать способ физической реализации репозитория. Описать средства и методы доступа к объектам репозитория.

3. Описать возможности графического языка, используемого для построения различных моделей разрабатываемой ИС. Перечислить виды диаграмм и описать их назначение.
4. Описать используемые подходы к организации коллективной разработки ИС и управлению командой проекта. Перечислить поддерживаемые виды и способы взаимодействия между членами команды разработчиков.
5. Описать возможности *CASE*-средств для автоматической генерации программного кода. Описать возможности быстрого макетирования (разработки макетов экранных и печатных форм) и прототипирования (разработки прототипов будущей ИС).
6. Описать возможности современных интегрированных сред разработки программного обеспечения. Описать способы применения ИСР в качестве *CASE*-средств автоматизации процесса разработки ПО.
7. Разработать документ, описывающий порядок применения *CASE*-технологии и *CASE*-средств для автоматизации процесса разработки ИС на всех стадиях жизненного цикла.

Варианты индивидуальных заданий

В качестве списка вариантов индивидуальных заданий используется перечень информационных систем из лабораторной работы № 2.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Информационные системы. Назначение, функции, области применения. Классификация ИС.
2. Архитектура ИС. Слои и звенья архитектуры. Клиент-серверная архитектура.
3. Моделирование информационных систем.
4. Виды моделей ИС: концептуальные, логические, физические.
5. Язык моделирования *UML*. Назначение, характеристики языка. Состав словаря языка *UML*.
6. Диаграммы *UML*. Виды диаграмм, их назначение.
7. Применение языка *UML* при создании ИС.
8. Жизненный цикл информационных систем.
9. Этапы жизненного цикла. Модели жизненного цикла.
10. Каскадная модель ЖЦ.
11. Инкрементная модель ЖЦ. Версии ИС. Прототип ИС.
12. Спиральная модель ЖЦ.
13. Современные методологии разработки ПО.
14. Методология *Microsoft Solutions Framework*. Модели и дисциплины *MSF*. Модель процессов. Фазы, вехи.
15. Методология *Rational Unified Process*. Итерации, фазы. Дисциплины *RUP*.
16. Гибкие методологии разработки (*Agile*).
17. Управление проектами. Проект как объект управления.
18. Программный проект. Особенности управления программным проектом.
19. Процесс разработки ПО. Спецификация. Проект. Проектирование.
20. Организация процесса разработки ПО.
21. Стандарты процессов жизненного цикла ИС.
22. Программные средства поддержки ЖЦ. *CASE*-технологии.
23. *CASE*-средства. Возможности *CASE*-средств. Особенности применения *CASE*-средств.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Брауде Э. Д. Технология разработки программного обеспечения: пер. с англ. – СПб.: Питер, 2004. – 655 с.
2. Амблер С. Гибкие технологии: экстремальное программирование и унифицированный процесс разработки – СПб.: Питер, 2005. – 412 с.
3. Ауэр К., Миллер Р. Экстремальное программирование: Постановка процесса: с первых шагов и до победного конца. – СПб.: Питер, 2004. – 368 с.
4. Вендров А. М. Проектирование программного обеспечения экономических информационных систем: учебник. - М.: Финансы и статистика, 2003. – 352 с.
5. Маккарти Д., Маккарти М. Правила разработки программного обеспечения: практ. руководство. – СПб.: Питер, 2007. – 240 с.
6. Орлов С. А. Технологии разработки программного обеспечения. Разработка сложных программных систем: учебник. – 3-е изд. – СПб.: ПИТЕР, 2004. – 527 с.
7. Михайлов А. А. Технологии проектирования информационных систем: метод. указания к курсовому проекту по дисциплине «Методология и технология проектирования информационных систем»; ЮРГПУ (НПИ) им. М. И. Платова. – Новочеркасск: Лик, 2016. – 13 с.
8. Мещеряков С. В., Иванов В. М. Эффективные технологии создания информационных систем. – СПб.: Политехника, 2005. – 309 с.
9. Технологии разработки программного обеспечения: учеб. пособие для сред. проф. образования, специальность «Программирование в компьютерных системах» / Р. М. Синецкий, С. А. Левшин, В. А. Есаулов; ЮРГПУ (НПИ) им. М. И. Платова. – Новочеркасск : ЮРГПУ (НПИ), 2014. – 84 с.