МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
Черниговский национальный технологический университет

ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторному практикуму
по дисциплине
“Проектирование программного обеспечения”
для студентов направления подготовки
6.050103 - “Программная инженерия”

Утверждено
на заседании кафедры
информационных технологий и программной инженерии

Протокол № 1 от 28.08.2016

Чернигов ЧНТУ 2017

Методичні вказівки до лабораторного практикуму з дисципліни „Проектування програмного забезпечення” для студентів напряму підготовки 6.050103 - “Програмна інженерія”./ Укл. А. М. Акименко, І. В. Богдан — Чернігів: ЧНТУ, 2017. — 49с. Рос. мовою.

Составители: Акименко Андрей Николаевич, кандидат физико-математических наук, доцент
Богдан Ирина Валентиновна, кандидат технических наук

Ответственный за выпуск: Литвинов Виталий Васильевич, заведующий кафедрой информационных технологий и программной инженерии, доктор технических наук, профессор

Рецензент: Скитер Игорь Семенович, кандидат физико-математических наук, доцент кафедры информационных технологий и программной инженерии Черниговского национального технологического университета

СОДЕРЖАНИЕ

ВСТУПЛЕНИЕ

Проектирование информационных систем всегда начинается с определения цели проекта. Основная задача любого успешного проекта заключается в том, чтобы на момент запуска системы и в течение всего времени ее эксплуатации можно было обеспечить:

- требуемую функциональность системы и степень адаптации к изменяющимся условиям ее функционирования;
- требуемую пропускную способность системы;
- требуемое время реакции системы на запрос;
- безотказную работу системы;
- простоту эксплуатации и поддержки системы;
- необходимую безопасность.

Проектирование информационных систем охватывает три основные области:

- проектирование объектов данных, которые будут реализованы в базе данных;
- проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
- учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

В реальных условиях проектирование — это поиск способа, который удовлетворяет требованиям функциональности системы средствами имеющихся технологий с учетом заданных ограничений.

К любому проекту предъявляется ряд абсолютных требований, например, максимальное время разработки проекта, максимальные денежные вложения в проект и т.д. Одна из сложностей проектирования состоит в том, что оно не является такой структурированной задачей, как анализ требований к проекту или реализация того или иного проектного решения.

1 ОБЩИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

1.1 Цель лабораторного практикума

Лабораторный практикум выполняется при изучении курса "Проектирование программного обеспечения" и имеет целью выработку у студентов навыков в трех направлениях:

- применение соответствующих методологий для проектирования и разработки информационных систем и программного обеспечения;
- применение языка UML для моделирования и проектирования информационных систем;
- применение соответствующего программного инструментария.

В "Общие указания" вынесены общие для выполнения всех лабораторных работ требования и правила.

1.2 Порядок выполнения лабораторных работ

Варианты индивидуального задания определяются преподавателем в соответствии со списком индивидуальных заданий, расположенном в разделе 9 данных Методических указаний.

Для выполнения всех лабораторных работ предлагается единый порядок, предусматривающий следующие шаги:

1. Ознакомиться с постановкой задачи и исходными данными.
2. Создать предлагаемую диаграмму или выполнить другое, указанное задание.
3. Сохранить результаты работы.
4. Составить отчет по проделанной работе.

1.3 Содержание отчета о выполнении лабораторных работ

Отчет оформляется по каждой лабораторной работе и состоит из следующих разделов:

1. Тема лабораторной работы.
2. Цель работы.
3. Индивидуальное задание.
4. Описание необходимых абстракций (элементов диаграмм) или хода выполнения лабораторной работы.
5. Разработанная диаграмма, созданное техническое задание или спецификация.

2 ЛАБОРАТОРНАЯ РАБОТА №1. ОПИСАНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1 Цель работы

Согласно варианту, выполнить описание предметной области проектируемой программной системы. Провести объектный анализ полученного описания и построить модель среды с помощью диаграммы потоков данных (анализ поведения системы) и диаграммы «сущность-связь» (анализ данных). Определить назначение проектируемой ИКС.

2.2 Теоретические сведения

2.2.1 Анализ предметной области

Этап анализа предполагает подробное исследование бизнес-процессов (функций, определенных на этапе выбора стратегии) и информации, необходимой для их выполнения (сущностей, их атрибутов и связей (отношений)). На этом этапе создается информационная модель системы.

Вся информация о системе формализуется и уточняется на этапе анализа. Особое внимание следует уделить полноте переданной информации, анализу информации на предмет отсутствия противоречий, а также поиску неиспользуемой вообще или дублирующейся информации.

Аналитики собирают и фиксируют информацию в двух взаимосвязанных формах:

- функции — информация о событиях и процессах, которые происходят в бизнесе;
- сущности — информация о вещах, имеющих значение для организации и о которых что-то известно.

Двумя классическими результатами анализа являются:

- иерархия функций, которая разбивает процесс обработки на составные части (что делается и из чего это состоит);
- модель «сущность-связь» (Entry Relationship model, ER-модель), которая описывает сущности, их атрибуты и связи (отношения) между ними.

Эти результаты являются необходимыми, но не достаточными. К достаточным результатам следует отнести диаграммы потоков данных.

Ниже мы рассмотрены наиболее часто применяемые методологии структурного анализа:

- диаграммы «сущность-связь» (Entity-Relationship Diagrams, ERD), которые служат для формализации информации о сущностях и их отношениях;

– диаграммы потоков данных (Data Flow Diagrams, DFD), которые служат для формализации представления функций системы.

2.2.2 Диаграммы «сущность-связь»

Диаграммы “сущность-связь” (ERD) предназначены для разработки моделей данных и обеспечивают стандартный способ определения данных и отношений между ними. Фактически с помощью ERD осуществляется детализация хранилищ данных проектируемой системы, а также документируются сущности системы и способы их взаимодействия, включая идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их отношений с другими объектами (связей).

Под сущностью (entity) понимается произвольное множество реальных или абстрактных объектов, каждый из которых обладает одинаковыми свойствами и характеристиками. В этом случае каждый рассматриваемый объект может являться экземпляром одной и только одной сущности, должен иметь уникальное имя или идентификатор, а также отличаться от других экземпляров данной сущности. Примерами сущностей могут быть: банк, клиент банка, компьютер, терминал. Каждая из сущностей может рассматриваться с различной степенью детализации и на различном уровне абстракции, что определяется конкретной постановкой задачи. Для графического представления сущностей используются специальные обозначения (рисунок 2.1).

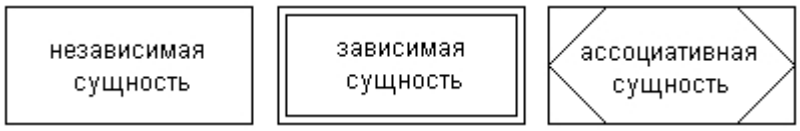

Рисунок 2.1 - Графические изображения для обозначения сущностей

Связь (relationship) определяется как отношение или некоторая ассоциация между отдельными сущностями. Примерами связей могут являться родственные отношения типа "отец-сын" или производственные отношения типа "начальник-подчиненный". Другой тип связей задается отношениями "иметь в собственности" или "обладать свойством". Различные типы связей графически изображаются в форме ромба с соответствующим именем данной связи (рисунок 2.2).

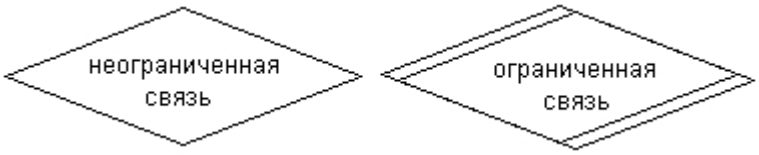

Рисунок 2.2 - Графические изображения для обозначения связей

2.2.3 Диаграммы потоков данных

Диаграммы потоков данных представляют собой информационную модель (DFD), основными компонентами которой являются различные потоки

данных, которые переносят информацию от одной подсистемы к другой. Каждая из подсистем выполняет определенные преобразования входного потока данных и передает результаты обработки информации в виде потоков данных для других подсистем.

Основными компонентами диаграмм потоков данных являются:

- внешние сущности,
- накопители данных или хранилища,
- процессы,
- потоки данных,
- системы/подсистемы.

Внешняя сущность представляет собой материальный объект или физическое лицо, которое может выступать в качестве источника или приемника информации. Примерами внешних сущностей могут служить: клиенты организации, заказчики, персонал, поставщики. Внешняя сущность обозначается прямоугольником с тенью (рисунок 2.3), внутри которого указывается ее имя. При этом в качестве имени рекомендуется использовать существительное в именительном падеже.

Рисунок 2.3 - Изображение внешней сущности на диаграмме потоков данных

Процесс представляет собой совокупность операций по преобразованию входных потоков данных в выходные в соответствии с определенным алгоритмом или правилом. Хотя физически процесс может быть реализован различными способами, наиболее часто подразумевается программная реализация процесса. Процесс на диаграмме потоков данных изображается прямоугольником с закругленными вершинами (рисунок 2.4), разделенным на три секции или поля горизонтальными линиями. Поле номера процесса служит для идентификации последнего. В среднем поле указывается имя процесса. В качестве имени рекомендовано использовать глагол в неопределенной форме с необходимыми дополнениями. Нижнее поле содержит указание на способ физической реализации процесса.

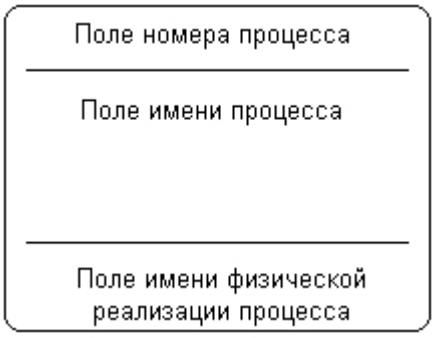

Рисунок 2.4 - Изображение процесса на диаграмме потоков данных

Накопитель данных или хранилище представляет собой абстрактное устройство или способ хранения информации, перемещаемой между процессами. Предполагается, что данные можно в любой момент поместить в накопитель и через некоторое время извлечь, причем физические способы помещения и извлечения данных могут быть произвольными. Накопитель данных на диаграмме потоков данных изображается прямоугольником с двумя полями (рисунок 2.5).

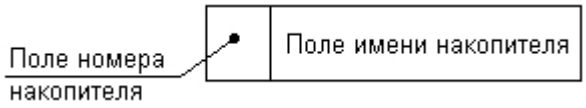

Рисунок 2.5 - Изображение накопителя на диаграмме потоков данных

Поток данных определяет качественный характер информации, передаваемой через некоторое соединение от источника к приемнику. Поток данных на диаграмме DFD изображается линией со стрелкой на одном из ее концов, при этом стрелка показывает направление потока данных. Каждый поток данных имеет свое собственное имя, отражающее его содержание.

Таким образом, информационная модель системы в нотации DFD строится в виде диаграмм потоков данных, которые графически представляются с использованием соответствующей системы обозначений.

2.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Описание предметной области.
3. Разработанные диаграммы потоков данных и «сущность-связь».
4. Назначение программной системы и описание её основных функций.
5. Выводы.

2.4 Контрольные вопросы

1. Цели проведения объектного анализа.
2. Назначение диаграммы «сущность-связь».
3. Основные элементы диаграммы «сущность-связь».
4. Назначение диаграммы потоков данных.
5. Основные элементы диаграммы потоков данных.

3 ЛАБОРАТОРНАЯ РАБОТА №2. ОФОРМЛЕНИЕ РЕЗУЛЬТАТОВ АНАЛИЗА ПРИ ПОМОЩИ ДИАГРАММ UML

3.1 Цель работы

Изучить правила оформления диаграммы вариантов использования и диаграммы анализа. Научится выделять особенности функционального поведения проектируемой системы.

3.2 Теоретические сведения

3.2.1 Построение диаграммы вариантов использования

Визуальное моделирование в UML можно представить как некоторый процесс поуровневого спуска от наиболее общей и абстрактной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы. Для достижения этих целей вначале строится модель в форме диаграммы вариантов использования (use case diagram), которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования.

Разработка диаграммы вариантов использования преследует цели:

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство или программа, которые могут служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актеру.

Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия ее внутренней структуры. В качестве такой сущности может выступать исходная система или любой другой элемент модели, который обладает собственным поведением.

Каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемую сущность или систему по запросу пользователя (актера), т. е. определяет способ применения этой сущности. Сервис, который инициализируется по запросу пользователя, представляет собой законченную последовательность действий. Это означает, что после того как система закончит обработку запроса пользователя, она должна возвратиться в исходное состояние, в котором готова к выполнению следующих запросов.

Между компонентами диаграммы вариантов использования могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров и вариантов. Один актер может взаимодействовать с несколькими вариантами использования. В этом случае этот актер обращается к нескольким сервисам данной системы. В свою очередь один вариант использования может взаимодействовать с несколькими актерами, предоставляя для всех них свой сервис. Следует заметить, что два варианта использования, определенные для одной и той же сущности, не могут взаимодействовать друг с другом, поскольку каждый из них самостоятельно описывает законченный вариант использования этой сущности.

В языке UML имеется несколько стандартных видов отношений между актерами и вариантами использования:

- отношение ассоциации (association relationship),
- отношение расширения (extend relationship),
- отношение обобщения (generalization relationship),
- отношение включения (include relationship),

При этом общие свойства вариантов использования могут быть представлены тремя различными способами, а именно с помощью отношений расширения, обобщения и включения.

Отношение ассоциации

Применительно к диаграммам вариантов использования ассоциация специфицирует семантические особенности взаимодействия актеров и вариантов использования в графической модели системы, то есть, это отношение устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. На диаграмме вариантов использования отношение ассоциации обозначается сплошной линией между актером и вариантом использования. Эта линия может иметь условные обозначения, такие как имя и кратность.

Кратность (multiplicity) ассоциации указывается рядом с обозначением компонента диаграммы, который является участником данной ассоциации, и характеризует количество экземпляров данного компонента, которые могут выступать в качестве элементов данной ассоциации. Применительно к диаграммам вариантов использования кратность имеет специальное обозначение в форме одной или нескольких цифр и символа звездочка.

Для диаграмм вариантов использования наиболее распространенными являются четыре основные формы записи кратности отношения ассоциации:

- целое неотрицательное число (включая 0). Предназначено для указания кратности, которая является строго фиксированной для элемента соответствующей ассоциации. В этом случае количество экземпляров актеров или вариантов использования, которые могут выступать в качестве элементов отношения ассоциации, в точности равно указанному числу;
- два целых неотрицательных числа, разделенные двумя точками. Данная запись соответствует нотации для множества или интервала целых чисел, которая применяется в некоторых языках программирования для обозначения границ массива элементов. Эту запись следует понимать как множество целых неотрицательных чисел, следующих в последовательно возрастающем порядке;
- два символа, разделенные двумя точками. При этом первый из них является целым неотрицательным числом или 0, а второй - специальным символом «*», который обозначает произвольное конечное целое неотрицательное число, значение которого неизвестно на момент задания соответствующего отношения ассоциации;
- единственный символ «*», который является сокращением записи интервала «0..*».

Если кратность отношения ассоциации не указана, то, по умолчанию, принимается значение равное 1.

Отношение расширения

Отношение расширения определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров. Отношение расширения является направленным и указывает, что применительно к отдельным примерам некоторого варианта использования должны быть выполнены конкретные условия, определенные для расширения данного варианта использования.

Отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой, направленной от того варианта использования, который является расширением для исходного варианта использования.

Отношение расширения отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования. Данное отношение включает в себя некоторое условие и ссылки на точки расширения в базовом варианте использования. Чтобы расширение имело место, должно быть выполнено определенное условие данного отношения.

Семантика отношения расширения определяется следующим образом. Если экземпляр варианта использования выполняет некоторую последовательность действий, которая определяет его поведение, и при этом имеется точка расширения на экземпляр другого варианта использования, которая является первой из всех точек расширения у исходного варианта, то проверяется условие

данного отношения. Если условие выполняется, исходная последовательность действий расширяется посредством включения действий экземпляра другого варианта использования.

Отношение обобщения

Отношение обобщения служит для указания того факта, что некоторый вариант использования А может быть обобщен до варианта использования В. В этом случае вариант А будет являться специализацией варианта В. При этом В называется предком или родителем по отношению А, а вариант А — потомком по отношению к варианту использования В. Следует подчеркнуть, что потомок наследует все свойства и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения. Графически данное отношение обозначается сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительский вариант использования.

Между отдельными актерами также может существовать отношение обобщения. Данное отношение является направленным и указывает на факт специализации одних актеров относительно других.

Отношение включения

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данное отношение является направленным бинарным отношением в том смысле, что пара экземпляров вариантов использования всегда упорядочена в отношении включения.

Семантика этого отношения определяется следующим образом. Когда экземпляр первого варианта использования в процессе своего выполнения достигает точки включения в последовательность поведения экземпляра второго варианта использования, экземпляр первого варианта использования выполняет последовательность действий, определяющую поведение экземпляра второго варианта использования, после чего продолжает выполнение действий своего поведения. При этом предполагается, что даже если экземпляр первого варианта использования может иметь несколько включаемых в себя экземпляров других вариантов, выполняемые ими действия должны закончиться к некоторому моменту, после чего должно быть продолжено выполнение прерванных действий экземпляра первого варианта использования в соответствии с заданным для него поведением.

3.2.2 Построение диаграммы анализа

Диаграмма анализа предназначена для описания происходящих в системе бизнес-процессов, модели поведения системы и ее элементов. Данная диаграмма является оптимальным средством для описания бизнес-процессов и их особенностей.

Диаграмма анализа представляет собой упрощенную версию диаграммы процессов Эрикссона-Пенкера, предназначенной для моделирования бизнес-процессов в сложных корпоративных системах.

Основными компонентами диаграммы анализа являются:

- бизнес-процесс,
- ресурс и информация,
- событие,
- выход,
- цель.

Бизнес-процесс

Бизнес-процесс представляет собой набор действий, направленных на получение конкретного результата для конкретного актера. Для каждого бизнес-процесса четко заранее определяются входы и выходы. Обозначение бизнес-процесса представлено на рисунке 3.1.

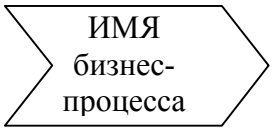

Рисунок 3.1 - Изображение бизнес-процесса на диаграмме анализа

Ресурс и информация

В процессе своей реализации бизнес-процесс использует ресурсы. В качестве ресурсов может выступать информация от внешних или внутренних источников, от других актеров или же от других бизнес-процессов.

В отличие от ресурсов, информация не используется бизнес-процессом, она несет информативный характер, объясняет или уточняет определенный нюанс.

Обозначение ресурса и информации представлено на рисунке 3.2.

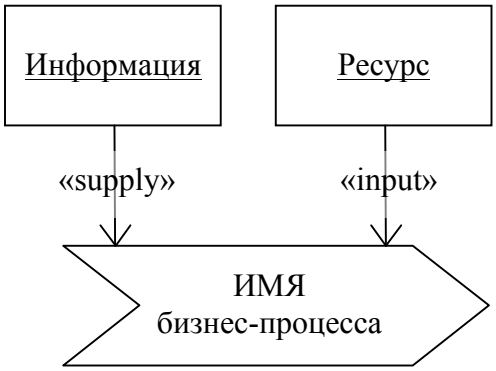

Рисунок 3.2 - Изображение ресурса и информации на диаграмме анализа

Событие

Событием является какой-либо объект, момент времени, дата, уведомление или какой-либо другой триггер, после срабатывания которого начинается

выполнение бизнес-процесса. Обозначение события представлено на рисунке 3.3.

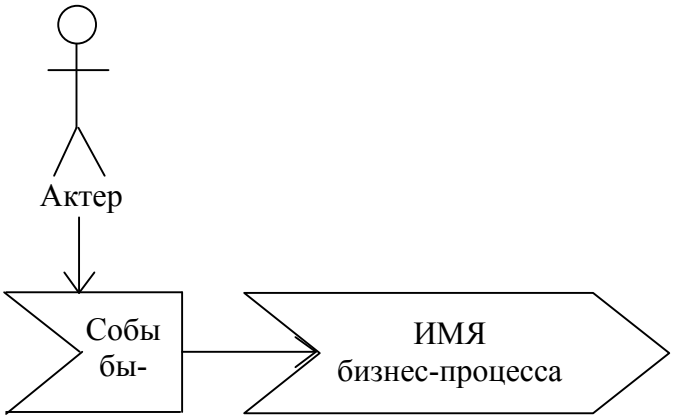

Рисунок 3.3 - Изображение ресурса на диаграмме анализа

Выход

В результате выполнения бизнес-процесса, как правило, формируется один или несколько выходных результатов. Выход может представлять собой физический объект (например, отчет), преобразование имеющихся ресурсов к новым условиям (ежедневное расписание или список) или общий результат деятельности (например, заказ).

Обозначение выхода представлено на рисунке 3.4.

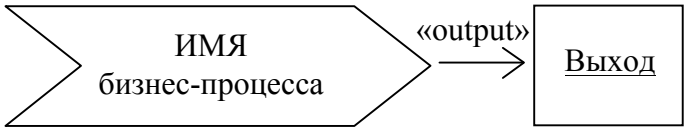

Рисунок 3.4 - Изображение выхода на диаграмме анализа

Цель

Любой бизнес-процесс имеет одну или несколько четко определенных целей. Именно цель является причиной организации и выполнения бизнес-процесса.

Обозначение цели представлено на рисунке 3.5.

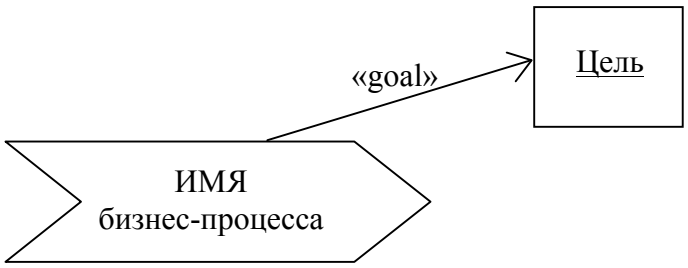

Рисунок 3.5 - Изображение цели на диаграмме анализа

3.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанные диаграммы вариантов использования.

3. Спецификация поведения элементов Use Case диаграммы вариантов использования.
4. Спецификация других элементов диаграммы.
5. Разработанные диаграммы анализа.
6. Выводы.

3.4 Контрольные вопросы

1. Назначение диаграммы вариантов использования.
2. Цели разработки диаграммы вариантов использования.
3. Элементы диаграммы вариантов использования. Актеры.
4. Элементы диаграммы вариантов использования. Отношения.
5. Элементы диаграммы вариантов использования. Use Case.
6. Цели разработки анализа.
7. Элементы диаграммы анализа. Бизнес-процессы.
8. Элементы диаграммы анализа. Ресурсы и информация.
9. Элементы диаграммы анализа. События.
10. Элементы диаграммы анализа. Выходы.
11. Элементы диаграммы анализа. Цели.

4 ЛАБОРАТОРНАЯ РАБОТА №3. ДИАГРАММА КЛАССОВ

4.1 Цель работы

Изучить правила оформления диаграммы классов. Научится разрабатывать статическую структуру модели системы в терминологии классов объектно-ориентированного программирования.

4.2 Теоретические сведения

4.2.1 Диаграмма классов

Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений.

Диаграмма классов состоит из множества элементов, которые в совокупности отражают декларативные знания о предметной области. Эти знания интерпретируются в базовых понятиях языка UML, таких как классы, интерфейсы и отношения между ними и их составляющими компонентами. При этом отдельные компоненты этой диаграммы могут образовывать пакеты для представления более общей модели системы. Если диаграмма классов является частью некоторого пакета, то ее компоненты должны соответствовать элементам этого пакета, включая возможные ссылки на элементы из других пакетов.

В общем случае пакет структурной статической модели может быть представлен в виде одной или нескольких диаграмм классов. Декомпозиция некоторого представления на отдельные диаграммы выполняется с целью удобства и графической визуализации структурных взаимосвязей предметной области. При этом компоненты диаграммы соответствуют элементам статической семантической модели. Модель системы, в свою очередь, должна быть согласована с внутренней структурой классов, которая описывается на языке UML.

Класс

Класс (class) в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции. В этих разделах могут указываться имя класса, атрибуты (переменные) и операции (методы).

Обязательным элементов обозначения класса является его имя. На начальных этапах разработки диаграммы отдельные классы могут обозначаться простым прямоугольником с указанием только имени соответствующего клас-

са. По мере проработки отдельных компонентов диаграммы, описания классов дополняются атрибутами и операциями.

Предполагается, что окончательный вариант диаграммы содержит наиболее полное описание классов, которые состоят из трех разделов или секций.

Отношения между классами

Кроме внутреннего устройства или структуры классов на соответствующей диаграмме указываются различные отношения между классами. При этом совокупность типов таких отношений фиксирована в языке UML и предопределена семантикой этих типов отношений. Базовыми отношениями или связями в языке UML являются:

- отношение зависимости (dependency relationship)
- отношение ассоциации (association relationship)
- отношение обобщения (generalization relationship)
- отношение реализации (realization relationship)

Каждое из этих отношений имеет собственное графическое представление на диаграмме, которое отражает взаимосвязи между объектами соответствующих классов.

Отношение зависимости

Отношение зависимости в общем случае указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов, которое не является отношением ассоциации, обобщения или реализации. Отношение зависимости используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели.

Отношение ассоциации

Отношение ассоциации соответствует наличию некоторого отношения между классами.

Наиболее простой случай данного отношения — бинарная ассоциация. Она связывает в точности два класса и, как исключение, может связывать класс с самим собой.

Отношение обобщения

Отношение обобщения является отношением между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком). Данное отношение может использоваться для представления взаимосвязей между пакетами, классами, вариантами использования и другими элементами языка UML.

Применительно к диаграмме классов данное отношение описывает иерархическое строение классов и наследование их свойств и поведения. При этом предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка.

Отношение реализации

Отношение реализации имеет место между двумя элементами модели в том случае, если один элемент (клиент) реализует поведение, заданное другим (поставщиком). Отношение реализации подразделяется на:

- отношение агрегации (aggregation relationship)
- отношение композиции (composition relationship)

Отношение агрегации

Отношение агрегации имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности.

Данное отношение имеет фундаментальное значение для описания структуры сложных систем, поскольку применяется для представления системных взаимосвязей типа "часть-целое". Раскрывая внутреннюю структуру системы, отношение агрегации показывает, из каких компонентов состоит система и как они связаны между собой.

Отношение композиции

Отношение композиции служит для выделения специальной формы отношения "часть-целое", при которой составляющие части в некотором смысле находятся внутри целого. Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого.

4.2.2 Рекомендации по построению диаграммы классов

Процесс разработки диаграммы классов занимает центральное место в ООАП сложных систем. От умения правильно выбрать классы и установить между ними взаимосвязи часто зависит не только успех процесса проектирования, но и производительность выполнения программы.

После разработки диаграммы классов процесс ООАП может быть продолжен в двух направлениях. С одной стороны, если поведение системы тривиально, то можно приступить к разработке диаграмм кооперации и компонентов. Однако для сложных динамических систем поведение представляет важнейший аспект их функционирования. Детализация поведения осуществляется последовательно при разработке диаграмм состояний, последовательности и деятельности.

4.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанные диаграммы классов.
3. Описание элементов диаграммы классов (включая отношения).
4. Выводы.

4.4 Контрольные вопросы

1. Назначение диаграммы классов.
2. Цели разработки диаграммы классов.
3. Элементы диаграммы классов. Классы.
4. Элементы диаграммы классов. Отношения.
5. Элементы диаграммы классов. Объекты.

5 ЛАБОРАТОРНАЯ РАБОТА №4. ДИАГРАММЫ ВЗАИМОДЕЙСТВИЯ

5.1 Цель работы

Изучить правила оформления диаграмм последовательности и кооперации. Изучить особенности взаимодействия объектов проектируемой системы.

5.2 Теоретические сведения

Функциональность элементов диаграммы вариантов использования отображается графически на диаграммах взаимодействия. Эти диаграммы содержат объекты и сообщения между объектами, которые показывают реализацию поведения.

5.2.1 Диаграмма последовательности

На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения. Одно — слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии. Графически каждый объект изображается прямоугольником и располагается в верхней части своей линии жизни.

Крайним слева на диаграмме изображается объект, который является инициатором взаимодействия. Правее изображается другой объект, который непосредственно взаимодействует с первым. Таким образом, все объекты на диаграмме последовательности образуют некоторый порядок, определяемый степенью активности этих объектов при взаимодействии друг с другом.

Второе измерение диаграммы последовательности — вертикальная временная ось, направленная сверху вниз. Начальному моменту времени соответствует самая верхняя часть диаграммы. При этом взаимодействия объектов реализуются посредством сообщений, которые посылаются одними объектами другим. Сообщения изображаются в виде горизонтальных стрелок с именем сообщения и также образуют, порядок по времени своего возникновения.

Линия жизни объекта

Линия жизни объекта (object lifeline) изображается пунктирной вертикальной линией, ассоциированной с единственным объектом на диаграмме последовательности. Линия жизни служит для обозначения периода времени, в течение которого объект существует в системе и, следовательно, может потенциально участвовать во всех ее взаимодействиях. Если объект существует в си-

стеме постоянно, то и его линия жизни должна продолжаться по всей плоскости диаграммы последовательности от самой верхней ее части до самой нижней.

Отдельные объекты, выполнив свою роль в системе, могут быть уничтожены, чтобы освободить занимаемые ими ресурсы. Для таких объектов линия жизни обрывается в момент его уничтожения. Для обозначения момента уничтожения объекта в языке UML используется специальный символ в форме латинской буквы "X".

Фокус управления

В процессе функционирования объектно-ориентированных систем одни объекты могут находиться в активном состоянии, непосредственно выполняя определенные действия или в состоянии пассивного ожидания сообщений от других объектов. Чтобы явно выделить подобную активность объектов, в языке UML применяется специальное понятие, получившее название фокуса управления (focus of control). Фокус управления изображается в форме вытянутого узкого прямоугольника, верхняя сторона которого обозначает начало получения фокуса управления объекта (начало активности), а ее нижняя сторона — окончание фокуса управления (окончание активности).

В отдельных случаях инициатором взаимодействия в системе может быть актер или внешний пользователь. В этом случае актер изображается на диаграмме последовательности самым первым объектом слева со своим фокусом управления. Чаще всего актер и его фокус управления будут существовать в системе постоянно, отмечая характерную для пользователя активность в инициировании взаимодействий с системой. При этом сам актер может иметь собственное имя либо оставаться анонимным.

Сообщения

Как было отмечено выше, цель взаимодействия в контексте языка UML заключается в том, чтобы специфицировать коммуникацию между множеством взаимодействующих объектов. Каждое взаимодействие описывается совокупностью сообщений, которыми участвующие в нем объекты обмениваются между собой. В этом смысле сообщение (message) представляет собой законченный фрагмент информации, который отправляется одним объектом другому. При этом прием сообщения инициирует выполнение определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено.

Таким образом, сообщения не только передают некоторую информацию, но и требуют или предполагают от принимающего объекта выполнения ожидаемых действий. Сообщения могут инициировать выполнение операций объектом соответствующего класса, а параметры этих операций передаются вместе с сообщением. На диаграмме последовательности все сообщения упорядочены по времени своего возникновения в моделируемой системе.

В таком контексте каждое сообщение имеет направление от объекта, который инициирует и отправляет сообщение, к объекту, который его получает.

Обычно сообщения изображаются горизонтальными стрелками, соединяющими линии жизни или фокусы управления двух объектов на диаграмме последовательности.

В языке UML каждое сообщение ассоциируется с некоторым действием, которое должно быть выполнено принявшим его объектом. При этом действие может иметь некоторые аргументы или параметры, в зависимости от конкретных значений которых может быть получен различный результат. Соответствующие параметры будет иметь и вызывающее это действие сообщение. Более того, значения параметров отдельных сообщений могут содержать условные выражения, образуя ветвление или альтернативные пути основного потока управления.

5.2.2 Диаграмма кооперации

Диаграмма кооперации предназначена для спецификации структурных аспектов взаимодействия. Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Прежде всего, на диаграмме кооперации изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее, как и на диаграмме классов, указываются ассоциации между объектами в виде различных соединительных линий. Дополнительно могут быть изображены динамические связи — потоки сообщений.

Таким образом, с помощью диаграммы кооперации можно описать полный контекст взаимодействий как своеобразный временной "срез" совокупности объектов, взаимодействующих между собой для выполнения определенной задачи или бизнес-цели программной системы.

Кооперация

Понятие кооперации (collaboration) является одним из фундаментальных понятий в языке UML. Оно служит для обозначения множества взаимодействующих с определенной целью объектов в общем контексте моделируемой системы. Цель самой кооперации состоит в том, чтобы специфицировать особенности реализации отдельных наиболее значимых операций в системе. Кооперация определяет структуру поведения системы в терминах взаимодействия участников этой кооперации.

Кооперация может быть представлена на двух уровнях:

- на уровне спецификации — показывает роли классификаторов и роли ассоциаций в рассматриваемом взаимодействии;
- на уровне примеров — указывает экземпляры и связи, образующие отдельные роли в кооперации.

Диаграмма кооперации уровня спецификации показывает роли, которые играют участвующие во взаимодействии элементы. Элементами кооперации на

этом уровне являются классы и ассоциации, которые обозначают отдельные роли классификаторов и ассоциации между участниками кооперации.

Диаграмма кооперации уровня примеров представляется совокупностью объектов (экземпляры классов) и связей (экземпляры ассоциаций). При этом связи дополняются стрелками сообщений. На данном уровне показываются только объекты, имеющие непосредственное отношение к реализации операции или классификатора.

5.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанные диаграммы последовательности.
3. Спецификация диаграмм последовательности.
4. Разработанные диаграммы кооперации уровня примеров.
5. Выводы.

5.4 Контрольные вопросы

1. Назначение диаграммы последовательности.
2. Особенности диаграммы последовательности.
3. Элементы диаграммы последовательности. Объекты.
4. Элементы диаграммы последовательности. Сообщения.
5. Диаграмма кооперации уровня спецификации.
6. Диаграмма кооперации уровня примеров.

6 ЛАБОРАТОРНАЯ РАБОТА №5. ДИАГРАММЫ ПОВЕДЕНИЯ

6.1 Цель работы

Изучить правила оформления диаграмм состояний и деятельности. Научится выделять в поведении элемента системы отдельные состояния. С помощью диаграммы деятельности научится отображать особенности алгоритмов, реализующих основные функции системы.

6.2 Теоретические сведения

6.2.1 Диаграмма состояний

Для моделирования поведения на логическом уровне в языке UML могут использоваться сразу несколько канонических диаграмм: состояний, деятельности, последовательности и кооперации, каждая из которых фиксирует внимание на отдельном аспекте функционирования системы. В отличие от других диаграмм диаграмма состояний описывает процесс изменения состояний только отдельного элемента модели (от отдельного экземпляра класса до всей системы в целом). При этом изменение состояния элемента системы может быть вызвано внешними воздействиями со стороны других подсистем или извне. Именно для описания реакции элемента модели на подобные внешние воздействия и используются диаграммы состояний.

Главное предназначение этой диаграммы — описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий. Системы, которые реагируют на внешние действия от других систем или от пользователей, иногда называют реактивными. Если такие действия инициируются в произвольные случайные моменты времени, то говорят об асинхронном поведении модели.

Состояние

Понятие состояния (state) является фундаментальным не только в метамодели языка UML, но и в прикладном системном анализе.

В языке UML под состоянием понимается абстрактный метакласс, используемый для моделирования отдельной ситуации, в течение которой имеет место выполнение некоторого условия. Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта.

Переход

Простой переход (simple transition) представляет собой отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния другим. Пребывание моделируемого объекта в первом состоянии может сопровождаться выполнением некоторых действий, а переход во второе состояние будет возможен после завершения этих действий, а также после удовлетворения некоторых дополнительных условий. В этом случае говорят, что переход срабатывает. До срабатывания перехода объект находится в предыдущем от него состоянии, называемым исходным состоянием, или в источнике, а после его срабатывания объект находится в последующем от него состоянии (целевом состоянии).

Переход осуществляется при наступлении некоторого события: окончания выполнения деятельности (do activity), получении объектом сообщения или приемом сигнала. На переходе указывается имя события. Кроме того, на переходе могут указываться действия, производимые объектом в ответ на внешние события при переходе из одного состояния в другое. Срабатывание перехода может зависеть не только от наступления некоторого события, но и от выполнения определенного условия, называемого сторожевым условием. Объект перейдет из одного состояния в другое в том случае, если произошло указанное событие и сторожевое условие приняло значение "истина".

6.2.2 Диаграмма деятельности

При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Традиционно для этой цели использовались блок-схемы или структурные схемы алгоритмов.

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления не деятельностей, а действий, и в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой, операции в предыдущем состоянии. Фактически, диаграммы деятельности можно считать частным случаем диаграмм состояний.

6.2.3 Рекомендации по построению диаграмм поведения

Диаграмма состояний

По своему назначению диаграмма состояний не является обязательным представлением в модели и как бы "присоединяется" к тому элементу, который

имеет нетривиальное поведение в течение своего жизненного цикла. Наличие у системы нескольких нетривиальных состояний, отличающихся от «исправен — неисправен», служит признаком необходимости построения диаграммы состояний. В качестве начального варианта диаграммы состояний, если нет очевидных соображений по поводу состояний объекта, можно воспользоваться этими суперсостояниями, рассматривая их как составные и детализируя их внутреннюю структуру по мере рассмотрения логики поведения объекта. При разработке диаграммы состояний нужно постоянно следить, чтобы объект в каждый момент мог находиться только в единственном состоянии. Если это не так, то данное обстоятельство может быть как следствием ошибки, так и неявным признаком наличия параллельности у поведения моделируемого объекта.

Диаграмма деятельности

Диаграммы деятельности играют важную роль в понимании процессов реализующих алгоритмы выполнения операций классов и потоков управления в моделируемой системе.

Содержание диаграммы деятельности во многом напоминает диаграмму состояний, хотя и не тождественно ей. В частности, эта диаграмма строится для отдельного класса, варианта использования, отдельной операции класса или целой подсистемы.

В случае типового проекта большинство деталей реализации действий могут быть известны заранее на основе анализа существующих систем или предшествующего опыта разработки систем-прототипов. Использование типовых решений может существенно сократить время разработки и избежать возможных ошибок при реализации проекта.

При разработке проекта новой системы, процесс функционирования которой основан на новых технологических решениях, ситуация более сложная. А именно, до начала работы над проектом могут быть неизвестны не только детали реализации отдельных деятельностей, но и само содержание этих деятельностей становится предметом разработки.

6.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанные диаграммы состояний.
3. Спецификация диаграмм состояний.
4. Разработанные диаграммы деятельности.
5. Выводы.

6.4 Контрольные вопросы

1. Назначение диаграммы состояний.
2. Особенности диаграммы состояний.
3. Элементы диаграммы состояний. Состояния.
4. Элементы диаграммы состояний. Переходы.
5. Диаграмма деятельности.

7 ЛАБОРАТОРНАЯ РАБОТА №6. ДИАГРАММА КОМПОНЕНТОВ

7.1 Цель работы

Изучить правила оформления диаграммы компонентов. Научится разрабатывать конкретную реализацию проекта в форме программного кода.

7.2 Теоретические сведения

7.2.1 Представление компонентов

Все рассмотренные ранее диаграммы относились к логическому уровню представления. Особенность логического представления заключается в том, что различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически, они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

Полный проект программной системы представляет собой совокупность моделей логического и физического представлений, которые должны быть согласованы между собой. В языке UML для физического представления моделей систем используются так называемые диаграммы реализации (implementation diagrams), которые включают в себя две отдельные канонические диаграммы: диаграмму компонентов и диаграмму развертывания.

Диаграмма компонентов описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код.

Диаграмма компонентов разрабатывается для следующих целей:

1. Визуализации общей структуры исходного кода программной системы.
2. Спецификации исполнимого варианта программной системы.
3. Обеспечения многократного использования отдельных фрагментов программного кода.
4. Представления концептуальной и физической схем баз данных.

Диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода. Одни компоненты могут существовать только на этапе компиляции программного кода, другие — на этапе его исполнения. Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов.

Компоненты

Для представления физических сущностей в языке UML применяется специальный термин — компонент (component). Компонент реализует некоторый набор интерфейсов и служит для общего обозначения элементов физического представления модели.

Компонент предоставляет организацию в рамках физического пакета ассоциированным с ним элементам модели. Как классификатор, компонент может иметь также свои собственные свойства, такие как атрибуты и операции.

Зависимости

Зависимость не является ассоциацией, а служит для представления только факта наличия такой связи, когда изменение одного элемента модели оказывает влияние или приводит к изменению другого элемента модели. Отношение зависимости на диаграмме компонентов изображается пунктирной линией со стрелкой, направленной от клиента (зависимого элемента) к источнику (независимому элементу).

Зависимости могут отражать связи модулей программы на этапе компиляции и генерации объектного кода. Применительно к диаграмме компонентов зависимости могут связывать компоненты и импортируемые этим компонентом интерфейсы, а также различные виды компонентов между собой.

Также на диаграмме могут быть представлены отношения зависимости между компонентами и реализованными в них классами. Эта информация имеет важное значение для обеспечения согласования логического и физического представлений модели системы.

7.2.2 Рекомендации по построению диаграммы компонентов

Разработка диаграммы компонентов предполагает использование информации, как о логическом представлении модели системы, так и об особенностях ее физической реализации. До начала разработки необходимо определиться с выбором языковой платформы и операционной системы.

После этого можно приступать к общей структуризации диаграммы компонентов. В первую очередь, необходимо решить, из каких физических частей (файлов) будет состоять программная система. На этом этапе следует обратить внимание на такую реализацию системы, которая обеспечивала бы не только возможность повторного использования кода за счет рациональной декомпозиции компонентов, но и создание объектов только при их необходимости.

После общей структуризации физического представления системы необходимо дополнить модель интерфейсами и схемами базы данных. При разработке интерфейсов следует обращать внимание на согласование различных частей программной системы. Включение в модель схемы базы данных предполагает спецификацию отдельных таблиц и установление информационных связей между таблицами.

7.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанная диаграмма компонентов.
3. Спецификация диаграммы компонентов.
4. Выводы.

7.4 Контрольные вопросы

1. Физическая модель программной системы.
2. Назначение диаграммы компонентов.
3. Цели разработки диаграммы компонентов.
4. Элементы диаграммы компонентов. Компоненты.
5. Элементы диаграммы компонентов. Зависимости.

8 ЛАБОРАТОРНАЯ РАБОТА №7. ДИАГРАММА РАЗВЕРТЫВАНИЯ

8.1 Цель работы

Изучить правила оформления диаграммы вариантов использования. Научится выделять особенности функционального поведения проектируемой системы.

8.2 Теоретические сведения

8.2.3 Диаграмма развертывания

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Этому есть несколько причин:

- сложные программные системы могут реализовываться в сетевом варианте на различных вычислительных платформах и технологиях доступа к распределенным базам данных;
- интеграция программной системы с Интернетом определяет необходимость решения дополнительных вопросов при проектировании системы, таких как обеспечение безопасности, устойчивости доступа к информации и т.д.
- технологии доступа и обработки данных в схеме "клиент-сервер" требует размещения больших баз данных в различных сегментах корпоративной сети, их резервного копирования для обеспечения необходимой производительности системы в целом.

Диаграмма развертывания (синоним — диаграмма размещения) применяется для представления общей конфигурации и топологии распределенной программной системы и содержит распределение компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений — маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками.

Итак, перечислим цели, преследуемые при разработке диаграммы развертывания:

- определить распределение компонентов системы по ее физическим узлам;
- показать физические связи между всеми узлами реализации системы на этапе ее исполнения;

– выявить узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности.

Узел

Узел (node) представляет собой некоторый физически существующий элемент системы, обладающий некоторым вычислительным ресурсом. В качестве вычислительного ресурса узла может рассматриваться наличие хотя бы некоторого объема электронной памяти и/или процессора. В последней версии языка UML понятие узла расширено и может включать в себя не только вычислительные устройства (процессоры), но и другие механические или электронные устройства.

Разрешено показывать на диаграмме развертывания узлы с вложенными изображениями компонентов. Важно помнить, что в качестве таких вложенных компонентов могут выступать только исполняемые компоненты.

Соединения

Кроме собственно изображений узлов на диаграмме развертывания указываются отношения между ними. В качестве отношений выступают физические соединения между узлами и зависимости между узлами и компонентами, изображения которых тоже могут присутствовать на диаграммах развертывания.

Соединения являются разновидностью ассоциации и изображаются отрезками линий без стрелок. Наличие такой линии указывает на необходимость организации физического канала для обмена информацией между соответствующими узлами. Характер соединения может быть дополнительно специфицирован примечанием, помеченным значением или ограничением.

Диаграммы развертывания могут иметь более сложную структуру, включающую вложенные компоненты, интерфейсы и другие аппаратные устройства.

8.2.4 Рекомендации по построению диаграммы развертывания

Разработка диаграммы развертывания начинается с идентификации всех аппаратных, механических и других типов устройств, которые необходимы для выполнения системой всех своих функций. В первую очередь специфицируются вычислительные узлы системы, обладающие памятью и/или процессором.

Дальнейшее построение диаграммы развертывания связано с размещением всех исполняемых компонентов диаграммы по узлам системы. Если отдельные исполняемые компоненты оказались не размещенными, то подобная ситуация должна быть исключена введением в модель дополнительных узлов, содержащих процессор и память.

При разработке простых программ, которые исполняются локально на одном компьютере, так же как и в случае диаграммы компонентов, необходимость в диаграмме развертывания может вообще отсутствовать. В более сложных ситуациях диаграмма развертывания строится для таких приложений, как:

- моделирование программных систем, реализующих технологию доступа к данным "клиент-сервер";
- моделирование неоднородных распределенных архитектур (корпоративные сети);
- моделирование системы со встроенными микропроцессорами, которые могут функционировать автономно.

Как правило, разработка диаграммы развертывания осуществляется на завершающем этапе ООАП, что характеризует окончание фазы проектирования физического представления. С другой стороны, диаграмма развертывания может строиться для анализа существующей системы с целью ее последующего анализа и модификации. При этом анализ предполагает разработку этой диаграммы на его начальных этапах, что характеризует общее направление анализа от физического представления к логическому.

8.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Разработанная диаграмма развертывания.
3. Спецификация диаграммы развертывания.
4. Выводы.

8.4 Контрольные вопросы

1. Назначение диаграммы развертывания.
2. Цели разработки диаграммы развертывания.
3. Элементы диаграммы развертывания. Узел.
4. Элементы диаграммы развертывания. Соединения.

9 ЛАБОРАТОРНАЯ РАБОТА №8. МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

9.1 Цель работы

Изучить существующие модели жизненного цикла программного обеспечения, оценить их достоинства и недостатки. Научится выбирать для каждой конкретной задачи наиболее подходящую для ее решения модель.

9.2 Теоретические сведения

Модель жизненного цикла программного обеспечения – это структура, которая определяет последовательность выполнения и взаимосвязи процессов, действий и задач на всем этапе выполнения проекта.

9.2.1 Каскадная модель

Каскадная модель предполагает строго последовательное (во времени) и однократное выполнение всех фаз проекта с жестким (детальным) предварительным планированием в контексте предопределенных или однажды и целиком определенных требований к программной системе.

9.2.2 Спиральная модель

Спиральная модель предполагает, что программное обеспечение создается не сразу, а итерационно с использованием метода прототипирования.

Прототитип – действующий компонент программного обеспечения, реализующий отдельные функции и внешние интерфейсы. Каждый виток спирали соответствует созданию фрагмента программного обеспечения, на нем уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы на следующий виток.

9.2.3 Инкрементная модель

Инкрементная модель предполагает разбиение жизненного цикла создаваемого проекта на последовательность итераций, каждая из которых напоминает “мини-проект”, включая все фазы жизненного цикла в применении к созданию меньших фрагментов функциональности, по сравнению с проектом, в целом.

Цель каждой итерации – получение работающей версии программной системы, включающей функциональность, определенную интегрированным содержанием всех предыдущих и текущей итерации.

9.2.4 V-образная модель

V-образная модель – это улучшенная версия классической каскадной модели. Здесь на каждом этапе происходит контроль текущего процесса, для того чтобы убедится в возможности перехода на следующий уровень. В этой модели тестирование начинается еще со стадии написания требований, причем для каждого последующего этапа предусмотрен свой уровень тестового покрытия.

В V-модели каждому этапу проектирования и разработки системы соответствует отдельный уровень тестирования. Здесь процесс разработки представлен нисходящей последовательностью в левой части условной буквы V, а стадии тестирования – на ее правом ребре. Соответствие этапов разработки и тестирования показано горизонтальными линиями.

9.2.5 Методология Scrum

Scrum – методология управления проектами, применяющаяся при необходимости гибкой разработки. В данной методологии акцент делается на качественном контроле процесса разработки.

Scrum – набор принципов, на которых строится процесс разработки, позволяющий в жестко фиксированные небольшие промежутки времени (спринты от 2 до 4 недель) предоставлять конечному пользователю работающее программное обеспечение с добавленными возможностями, для которых определен наибольший приоритет.

9.2.6 Экстремальное программирование

Экстремальное программирование – это методология организации разработки программ для небольших и средних по размеру команд разработчиков, занимающихся созданием программного обеспечения в условиях неясных или быстро меняющихся требований.

Разработка ведется итерациями, длительность которых составляет 2-3 недели, максимум месяц. За одну итерацию выполняются все запланированные пользовательские истории.

Пользовательская история – это 1-2 абзаца информации, на основании которой создается модуль, пишется самими пользователями или представителями заказчика, которые в XP являются частью команды.

9.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Выбранная модель жизненного цикла программного обеспечения и ее описание с указанием достоинств и недостатков.
3. Описание того, почему именно данная модель является наиболее подходящей для создания выбранной ИКС.
4. Выводы.

9.4 Контрольные вопросы

1. Классификация моделей жизненного цикла программного обеспечения.
2. Суть каждой из существующих моделей жизненного цикла программного обеспечения.
3. Достоинства и недостатки каждой из существующих моделей жизненного цикла программного обеспечения.

10 ЛАБОРАТОРНАЯ РАБОТА №9. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

10.1 Цель работы

Ознакомиться с правилами написания технического задания. Научится самостоятельно составлять техническое задание для каждой конкретной задачи.

10.2 Теоретические сведения

Техническое задание – это документ, в котором формулируются основные цели разработки, требования к программному продукту, определяются сроки и этапы разработки и регламентируется процесс приемно-сдаточных испытаний.

В формулировании технического задания участвуют как представители заказчика, так и представители исполнителя.

10.2.1 Разделы технического задания

Техническое задание создается в соответствии со стандартом ГОСТ 19.201–78, согласно которому оно должно содержать следующие разделы:

1. Введение – раздел должен включать наименование и краткую характеристику области применения программы или программного продукта.

2. Основания для разработки – раздел должен содержать наименование документа, на основании которого ведется разработка, организации, утвердившей данный документ, и наименование или условное обозначение темы разработки.

3. Назначение разработки – раздел должен содержать описание функционального и эксплуатационного назначения программного продукта с указанием категорий пользователей.

4. Требования к программе или программному изделию – раздел должен включать требования к функциональным характеристикам, требования к надежности, условия эксплуатации, требования к составу и параметрам технических средств, требования к информационной и программной совместимости, требования к маркировке и упаковке, требования к транспортированию и хранению, специальные требования.

5. Требования к программной документации – раздел указывает на необходимость наличия руководства программиста, руководства пользователя, руководства системного программиста, пояснительной записки и т.п.

6. Технико-экономические показатели – раздел содержит ориентировочную экономическую эффективность и экономические преимущества по сравнению с существующими аналогами.

7. Стадии и этапы разработки – раздел содержит стадии разработки, этапы и содержание работ с указанием сроков разработки и исполнителей.

8. Порядок контроля и приемки – раздел содержит виды испытаний и общие требования к приемке работы.

10.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Техническое задание к создаваемой согласно варианту ИКС.
4. Выводы.

10.4 Контрольные вопросы

1. Что такое техническое задание и каким стандартом оно регламентируется.
2. Факторы, определяющие характеристики разрабатываемого программного обеспечения.
3. Действия, выполняемые перед созданием технического задания.
4. Обязательные разделы технического задания.

11 ЛАБОРАТОРНАЯ РАБОТА №10. СПЕЦИФИКАЦИЯ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

11.1 Цель работы

Ознакомиться с правилами создания спецификации требований к программному обеспечению. Научится самостоятельно составлять спецификацию для каждой конкретной задачи.

11.2 Теоретические сведения

Спецификация требований к программному обеспечению (SRS) - это спецификация для определенного программного изделия, программы или набора программ, которые выполняют определенные функции в специфической среде. SRS может составляться одним или более представителями поставщика, одним или более представителями заказчика, или обоими.

11.2.1 Разделы спецификации

Основными разделами, содержащимися в спецификации требований к программному обеспечению, являются:

1. Введение – раздел должен содержать краткий обзор SRS.
2. Общее описание – раздел должен содержать общие факторы, которые влияют на программное изделие и требования, предъявляемые к нему. Этот раздел не устанавливает конкретные требования. Вместо этого, он обеспечивает предварительные сведения о тех требованиях, которые подробно определяются в разделе 3, и делает их более простыми для понимания.
3. Специфические требования – раздел должен содержать все требования к программному обеспечению на уровне детализации, достаточной, чтобы дать проектировщикам возможность разработать систему, удовлетворяющую этим требованиям, а тестировщикам - убедиться, что система удовлетворяет этим требованиям.
4. Вспомогательная информация – раздел должен включать содержание, алфавитный указатель и приложения.

11.3 Содержание отчета

1. Наименование и цель работы, номер варианта.
2. Спецификация требований к создаваемой согласно варианту ИКС.
4. Выводы.

11.4 Контрольные вопросы

1. Что такое спецификация требований к программному обеспечению.
2. Основные вопросы, которые должны рассматривать составители спецификаций.

3. Основные характеристики правильно созданной SRS.
4. Основные разделы SRS.

12 СПИСОК ИНДИВИДУАЛЬНЫХ ВАРИАНТОВ ЗАДАНИЙ СТУДЕНТОВ

1. Разработать модель ИКС библиотеки (актеры – руководитель, библиотекарь, читатель)
2. Разработать модель ИКС рекламной фирмы (актеры – руководитель, сотрудник по работе с клиентами, клиент)
3. Разработать модель ИКС видеосалона (актеры – руководитель, сотрудник, клиент)
4. Разработать модель ИКС магазина парфюмерии (актеры – руководитель, сотрудник, клиент)
5. Разработать модель ИКС ресторана (актеры – руководитель, официант, клиент)
6. Разработать модель ИКС организации по работе с абитуриентами (актеры – руководитель, сотрудник организации, абитуриент)
7. Разработать модель ИКС средней школы (актеры – директор, преподаватель, ученик)
8. Разработать модель ИКС провайдера Интернет (актеры – руководитель, сотрудник по работе с клиентами, клиент)
9. Разработать модель ИКС работы военкомата (актеры – руководитель, сотрудник по работе с призывниками, призывник)
10. Разработать модель ИКС работы центра занятости (актеры – руководитель, сотрудник по работе с безработными, безработный)
11. Разработать модель ИКС системы охраны предприятия (актеры – руководитель предприятия, сотрудник предприятия, охранник)
12. Разработать модель ИКС работы университета (актеры – ректор, декан, студент)
13. Разработать модель ИКС супермаркета (актеры – директор, кассир, покупатель)
14. Разработать модель ИКС чемпионата по хоккею (актеры – сотрудник по работе с хоккеистами, хоккеист, глава федерации хоккея)
15. Разработать модель ИКС олимпийских игр (актеры – болельщик, сотрудник по работе с болельщиками, руководитель олимпийского комитета)
16. Разработать модель ИКС зоопарка (актеры – посетитель, директор, кассир)
17. Разработать модель ИКС театра (актеры – актер, директор, администратор)
18. Разработать модель ИКС страхового агентства (актеры – клиент, директор, юрист)
19. Разработать модель ИКС свадебного салона (актеры – директор, сотрудник по работе с клиентами, клиент)
20. Разработать модель ИКС туристической фирмы (актеры – директор, сотрудник по работе с клиентами, клиент)

21. Разработать модель ИКС парикмахерской (актеры – руководитель, парикмахер, клиент)

22. Разработать модель ИКС пиццерии (актеры – руководитель, официант, клиент)

23. Разработать модель ИКС аукционного дома (актеры – руководитель, сотрудник, аукционист)

24. Разработать модель ИКС автопарк (актеры – директор, клиент, водитель)

25. Разработать модель ИКС салона по продаже мобильных телефонов (актеры – директор, продавец-консультант, клиент)

26. Разработать модель ИКС кинологического клуба (актеры – руководитель, кинолог, владелец животного)

27. Разработать модель ИКС детского сада (актеры – директор, воспитатель, родитель)

28. Разработать модель ИКС управления программными проектами (актеры – руководитель фирмы, разработчик, менеджер проектов)

29. Разработать модель ИКС командной разработки курсовых проектов (актеры – ректор, преподаватель, заведующий кафедрой)

30. Разработать модель ИКС Министерства образования (актеры – министр образования, сотрудник министерства, президент)

31. Разработать модель ИКС вокзала (актеры – начальник, машинист, пассажир)

32. Разработать модель ИКС благотворительного фонда (актеры – руководитель, меценат, сотрудник по работе с клиентами)

33. Разработать модель ИКС учета ГАИ (актеры – адвокат, гаишник, свидетель правонарушения)

34. Разработать модель ИКС работы коммунального предприятия (актеры – директор, диспетчер, житель)

35. Разработать модель ИКС ремонтной мастерской (актеры – директор, мастер, клиент)

36. Разработать модель ИКС троллейбусного управления (актеры – начальник, водитель, технический работник)

37. Разработать модель ИКС жилищно-комунального предприятия (актеры – начальник, диспетчер, сантехник)

38. Разработать модель ИКС курьерского агентства (актеры – начальник, заказчик курьерских услуг, сотрудник по работе с клиентами)

39. Разработать модель ИКС агентства по уходу за людьми преклонного возраста (актеры – руководитель, человек преклонного возраста, психолог)

40. Разработать модель ИКС аптечного фонда (актеры – начальник, клиент, фармацефт)

41. Разработать модель ИКС предоставления услуг мобильного оператора (актеры – начальник, потребитель, технический работник)

42. Разработать ИКС кинотеатра (актеры – директор, киноман, кассир)

43. Разработать модель ИКС планетария (актеры – руководитель, экскурсовод, посетитель)

44. Разработать модель ИКС парка развлечений (актеры – директор, посетитель, кассир)

45. Разработать модель ИКС магазина бытовой техники (актеры – директор, клиент, продавец)

46. Разработать модель ИКС банка (актеры – глава, кассир, клиент)

47. Разработать модель ИКС кафе (актеры – начальник, официант, посетитель)

48. Разработать модель ИКС городских электрических сетей (актеры – начальник, электрик, диспетчер)

49. Разработать модель ИКС ЗАГСА (актеры – руководитель, сотрудник отдела приема заявлений, желающий женится)

50. Разработать модель ИКС фитнес-клуба (актеры – руководитель, тренер, клиент)

51. Разработать модель ИКС работы отдела кредитования банка (актеры – глава, клиент, сотрудник, оформляющий кредиты)

52. Разработать модель ИКС видеопроката (актеры – директор, сотрудник, клиент)

53. Разработать модель ИКС зала игровых автоматов (актеры – директор, техник-настройщик, игрок)

54. Разработать модель ИКС работы отдела кадров электромеханического завода (актеры – начальник отдела кадров, сотрудник отдела кадров, представитель биржи труда)

55. Разработать модель ИКС фабрики по производству музыкальных инструментов (актеры – руководитель, сотрудник, клиент)

56. Разработать модель ИКС маркетинговой фирмы (актеры – маркетолог, сотрудник по работе с клиентами, клиент)

57. Разработать модель ИКС лыжной базы (актеры – директор, тренер, технический работник)

58. Разработать модель ИКС авиакомпании (актеры – директор, пилот, пассажир)

59. Разработать модель ИКС агентства недвижимости (актеры – руководитель, риэлтер, юрист)

60. Разработать модель ИКС магазина спортивных товаров (актеры – руководитель, продавец-консультант, клиент)

61. Разработать модель ИКС миграционной службы (актеры – начальник, юрист, мигрант)

62. Разработать модель ИКС учета пациентов поликлиники (актеры – больной, работник регистратуры, доктор)

63. Разработать модель ИКС склада торговой фирмы (актеры – грузчик, кладовщик, водитель транспорта по перевозке продукции со/на склад торговой фирмы)

64. Разработать модель ИКС мебельного магазина (актеры – директор, продавец-консультант, клиент)

65. Разработать модель ИКС предприятия по созданию мебели (актеры – руководитель, клиент, менеджер отдела продаж)

66. Разработать модель ИКС ателье по пошиву пальто (актеры – директор, клиент, швея)

67. Разработать модель ИКС прачечной (актеры – клиент, прачка, сотрудник службы доставки)

68. Разработать модель ИКС химчистки (актеры – клиент, сотрудник, работающий с чистящим аппаратом, сотрудник службы доставки)

69. Разработать модель ИКС автосервиса (актеры – руководитель, автомеханик, клиент)

70. Разработать модель ИКС лизинга автомобилей (актеры – руководитель, клиент, сотрудник по работе с клиентами)

71. Разработать модель ИКС магазина товаров для туризма и отдыха (актеры – директор, продавец-консультант, клиент)

72. Разработать модель ИКС деревообрабатывающего предприятия (актеры – заказчик, столяр, сотрудник по работе с клиентами)

73. Разработать модель ИКС книжного магазина (актеры – директор, продавец-консультант, клиент)

74. Разработать модель ИКС букмекерской конторы (актеры – начальник, сотрудник, отвечающий за прием ставок, желающий сделать ставку)

75. Разработать модель ИКС травматологического пункта (актеры – медицинская сестра, больной, врач)

76. Разработать модель ИКС центра тестирования выпускников (актеры – родитель выпускника, сотрудник центра, выпускник)

77. Разработать модель ИКС клиники пластической хирургии (актеры – заведующий, пластический хирург, желающий сделать пластическую операцию)

78. Разработать модель ИКС центра помощи ветеранам (актеры – руководитель, сотрудник центра, ветеран)

79. Разработать модель ИКС центра помощи многодетным семьям (актеры – руководитель, сотрудник центра, представитель многодетной семьи)

80. Разработать модель ИКС детского садика (актеры – руководитель, повар, родитель)

81. Разработать модель ИКС исторического музея (актеры – руководитель, сотрудник, посетитель)

82. Разработать модель ИКС работы паспортного стола (актеры – начальник, клиент, сотрудник по приему заявок)

83. Разработать модель ИКС работы приемной городского главы (актеры – руководитель, сотрудник приемной, гражданин)

84. Разработать модель ИКС работы магазина компьютерной техники (актеры – директор, продавец-консультант, клиент)

85. Разработать модель ИКС организации защиты животных (актеры – руководитель, ветеринар, владелец животного)

86. Разработать модель ИКС работы пенсионного фонда (актеры – начальник, сотрудник по работке с клиентами, пенсионер)

87. Разработать модель ИКС агентства знакомств (актеры – руководитель, сотрудник по работе с клиентами, клиент)

88. Разработать модель ИКС работы Ледовой арены (актеры – руководитель, охранник, посетитель)

89. Разработать модель ИКС работы парка развлечений (актеры – руководитель, кассир, посетитель)

90. Разработать модель ИКС работы речного вокзала (актеры – пассажир, капитан, кассир)

РЕКОМЕНДОВАННАЯ ЛИТЕРАТУРА

1. А. Леоненков. Самоучитель UML. Эффективный инструмент моделирования информационных систем. – BHV-Санкт-Петербург, 2001.- 304с.
2. Гради Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2-е изд.- М. : "Бином", 1999 г.- 560с.
3. Джеймс Рамбо, Айвар Якобсон, Грэди Буч. UML. Специальный справочник. – Питер, 2002. – 656с.
4. М.Фаулер, К.Скотт. UML в кратком изложении. Применение стандартного языка объектного моделирования. – М.: Мир, 1999. – 192с.
5. С.А.Трофимов. CASE-технологии. Практическая работа в Rational Rose. – Бином, 2002.-272с.
6. Терри Кватрани. Rational Rose 2000 и UML. Визуальное моделирование. – ДМК, 2001.- 176с.
7. Кент Бек. Экстримальное программирование: разработка через тестирование. – Питер, 2003.- 224с.
8. Стандарт IEEE 830-1998 [Электронный ресурс]. – Режим доступа: https://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwjN3NDy09zQAhXFBZoKHRhtCJMQFggZMAA&url=http%3A%2F%2Fkspt.icc.spbstu.ru%2Fmedia%2Ffiles%2F2009%2Fcourse%2Fse%2FIEEE-830-1998_RU.doc&usg=AFQjCNExgFNCt_mHy_r4_l5yCDhHgD1jQQ&sig2=gxBP57d1brebKeTFY4z7Dg&bvm=bv.139782543,d.bGs&cad=rjt.
9. Язык UML. Руководство пользователя. – ДМК, 2000. – 432с.