МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСТЕТ»
(ДГТУ)

Кафедра «Информационные технологии»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ»

Ростов-на-Дону
ДГТУ
2020

УДК 372.8:004
Составители: М. В. Привалов

Методические указания для выполнения лабораторных работ по дисциплине «Инструментальные средства проектирования информационных систем». - Ростов-на-Дону: Донской гос. техн. ун-т, 2020. – 14 с.

Рассматриваются современные подходы и технологии разработки программного обеспечения для Web-ориентированных и распределённых информационных систем.

Предназначены для студентов направления 09.04.02 «Информационные системы и технологии» всех форм обучения.

УДК 372.8:004

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

Ответственный за выпуск зав. кафедрой «Информационные технологии»,
д-р техн. наук, профессор Б.В. Соболь

В печать ___.___. 20___г.
Формат 60×84/16. Объем___усл.п.л.
Тираж ___экз. Заказ №___.

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина,1

Лабораторная работа №1
«ОПРЕДЕЛЕНИЕ КОНТЕКСТА И ГРАНИЦ ПРОДУКТА»

Цель работы: научиться определять функциональные границы продуктов и контекст.
Форма отчета: продемонстрировать выполненное задание преподавателю

Методические указания:

Теоретические сведения.

Для создания эффективных программных решений необходимо обеспечить решение всех ключевых проблем в процессе производства, для которого создается программный продукт. С этой целью определяется область применения продукта, целевая аудитория, проблемы заказчика и конечных пользователей, которые жизненно необходимо решить. Затем выделяется перечень свойств разрабатываемого продукта.

Выполненная работа представляется в виде двух документа Vision & Scope (Видение и контекст продукта).

Видение продукта раскрывает идею, позволяющую решить проблемы заказчика и получить прибыль за счёт внедрения информационной системы.

Контекст продукта определяет область его применения и границы, заданные функциональными и нефункциональными требованиями.

Существует несколько возможных шаблонов формирования документа Vision&Scope, а его форма зависит от применяемого подхода и организации жизненного цикла разработки программного обеспечения.

В данной работе предлагается взять за основу вариант, принятый в методологии разработки Microsoft Solutions Framework (MSF). Пример прилагается.

На основании свойств продукта формулируются требования к программному обеспечению, которые принято разделять на функциональные и нефункциональные.

Задание

В соответствии с индивидуальным заданием определите функциональные границы продукта и представьте их в виде документа Vision & Scope.

Индивидуальное задание

Индивидуальное задание согласовывается с преподавателем. Допускается в качестве индивидуального задания использовать тему своего дипломного проекта или одно из перечисленных ниже заданий.

1. Спроектировать компьютерную подсистему учета Интернет-услуг
2. Спроектировать подсистему учета ресурса воздушных судов
3. Спроектировать компьютерную подсистему начисления и учета страховых выплат клиентам в условиях страховой компании
4. Спроектировать подсистему учета и анализа экономической деятельности в условиях агентства недвижимости
5. Спроектировать подсистему формирования и обработки договоров добровольного страхования наземного автотранспорта
6. Спроектировать подсистему обработки торговых агентов
7. Спроектировать подсистему планирования и учета чартеров
8. Спроектировать подсистему формирования и учета счетов-заказов для туристических агентств
9. Спроектировать подсистему бронирования билетов в кинотеатр
10. Спроектировать подсистему расчета себестоимости для рыбного хозяйства
11. Спроектировать подсистему учета потребительских кредитов
12. Спроектировать подсистему реализации и движения горюче-смазочных материалов в

условиях сети АЗС
13. Спроектировать подсистему учета продаж товаров
14. Спроектировать подсистему учета сбора и реализации зерновых культур
15. Спроектировать подсистему учета убытков автогражданской ответственности
16. Спроектировать подсистему учета выплат за услуги газоснабжения для населения
17. Спроектировать подсистему расчета арендной платы в условиях коммунального предприятия
18. Спроектировать подсистему учета движения грузового подвижного железнодорожного транспорта в условиях металлургического завода
19. Спроектировать подсистему учета сырья и материалов
20. Спроектировать подсистему приема электронных коммунальных платежей в условиях банка
21. Спроектировать подсистему учета и планирования ремонтных работ
22. Спроектировать подсистему учета технического состояния компьютерного оборудования

Лабораторная работа №2
«ОПРЕДЕЛЕНИЕ СВОЙСТВ ПРОДУКТА»

Цель работы: научиться определять свойства, необходимые для решения проблем заказчика, формулировать требования.
Форма отчета: продемонстрировать выполненное задание преподавателю

Методические указания:

Теоретические сведения.

После определения области применения продукта, целевой аудитории, проблемы заказчика и конечных пользователей, которые жизненно необходимо решить, выделяется перечень свойств разрабатываемого продукта. Для этого используется документ Product Features (Свойства продукта). Такой документ пришёл в проектирование систем из Rational Unified Process (RUP). Он отражает помимо концепции создания продукта его свойства, функциональные и основные нефункциональные требования, а также соответствие функций системы требованиям заказчика.

На основании свойств продукта формулируются требования к программному обеспечению, поэтому его составление требует внимания и тщательной проверки покрытия потребностей заинтересованных лиц запланированными свойствами продукта.

Задание.

В соответствии с индивидуальным заданием предыдущей работы:

1. Выделите свойства продукта и разработайте соответствующий документ, используя лекционный пример Product Features в качестве шаблона.
2. Сформулируйте основные функциональные и нефункциональные требования к программному обеспечению (без детализации сценариев Use Case).
3. Проверьте соответствие выделенных свойств продукта требованиям заказчика.

Лабораторная работа №3
«ОПИСАНИЕ ТЕХНИЧЕСКИХ ТРЕБОВАНИЙ К АРХИТЕКТУРЕ»

Цель работы: научиться определять и описывать технические требования к системам.

Форма отчета: продемонстрировать выполненное задание преподавателю

Методические указания:

Теоретические сведения.

После определения видения и контекста, а также проектирования свойств продукта (документы Vision&Scope и Product Features) уже выяснены высокоуровневые функциональные требования и основные нефункциональные требования к системе. После этого производится описание архитектуры в соответствии с какой-либо методикой (4 + 1, Rational Unified Process и др.)

Любые методики представления архитектуры системы предполагают включение её описаний с различных сторон, так называемых архитектурных представлений. Прежде чем применять модель Кратчена и другие ей подобные, необходимо определить технические требования: архитектурные цели и задачи, ограничения, атрибуты качества, стек технологий.

Задание.

Архитектурные цели, технические ограничения и атрибуты качества необходимо описать в виде таблиц, приведенных ниже.

Таблица 3.1 – Архитектурные цели и задачи

#	Архитектурные цели	Описание	Как выполняется?

Таблица 3.2 – Технические ограничения

#	Ограничение	Описание	Как влияет на архитектуру?

Таблица 3.3 – Атрибуты качества

#	Атрибут	Описание	Как выполняется?

В колонке «Описание» фактически указывается формулировка требования к заданному атрибуту/цели/ограничению. При описании требований рекомендуется использовать общепринятые характеристики и их единицы измерения.

Контрольные вопросы

1. Что такое архитектурные требования? Что они описывают?
2. Что такое атрибуты качества?
3. Чем атрибуты качества отличаются от функциональных и нефункциональных требований?
4. Как можно удовлетворить тот или иной атрибут качества?
5. Что такое технические ограничения?
6. В чём отличие технических ограничений от бизнес-ограничений?

Лабораторная работа №4
«ОПИСАНИЕ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ»

Цель работы: научиться применять нотацию IDEF0 для выделения высокоуровневых функций системы и описания функциональных требований

Форма отчета: продемонстрировать выполненное задание преподавателю

Методические указания:

Теоретические сведения.

После определения видения и контекста, а также проектирования свойств продукта (документы Vision&Scope и Product Features) уже выяснены высокоуровневые функциональные требования и основные нефункциональные требования к системе. После этого производится описание архитектуры в соответствии с какой-либо методикой (IDEF, 4 + 1, Rational Unified Process и др.)

Любые методики представления архитектуры системы предполагают включение её описаний с различных сторон, так называемых архитектурных представлений. В подходе IDEF, позволяющем выполнить функциональную декомпозицию системы, строится IDEF0-диаграмма, при этом на ней каждая функция представляет собой блок диаграммы (рис. 4.1).

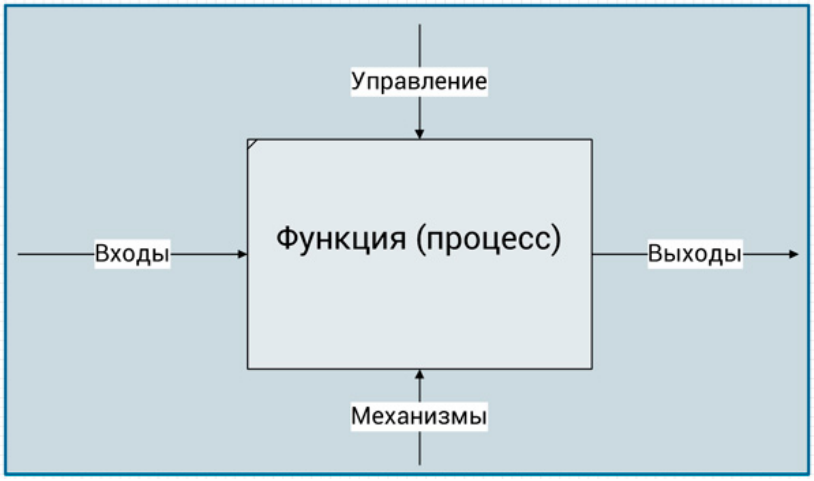

Рис. 4.1 – Функциональный блок IDEF0

Как правило, всю систему изначально представляют одним функциональным блоком, который иногда называют «моделью чёрного ящика» (рис. 4.2).

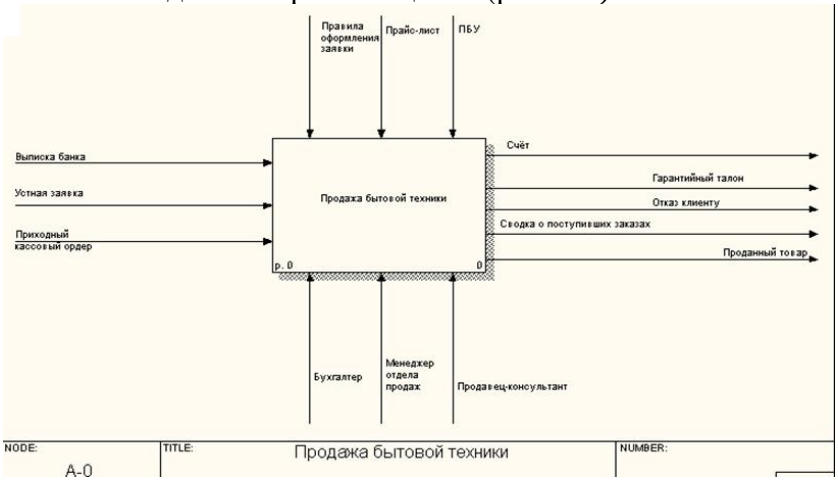

Рис. 4.2 – Представление системы

Для выделения основных функций системы выполняется первый уровень декомпозиции. Он позволяет получить в результате список высокоуровневых функций (рис 4.3).

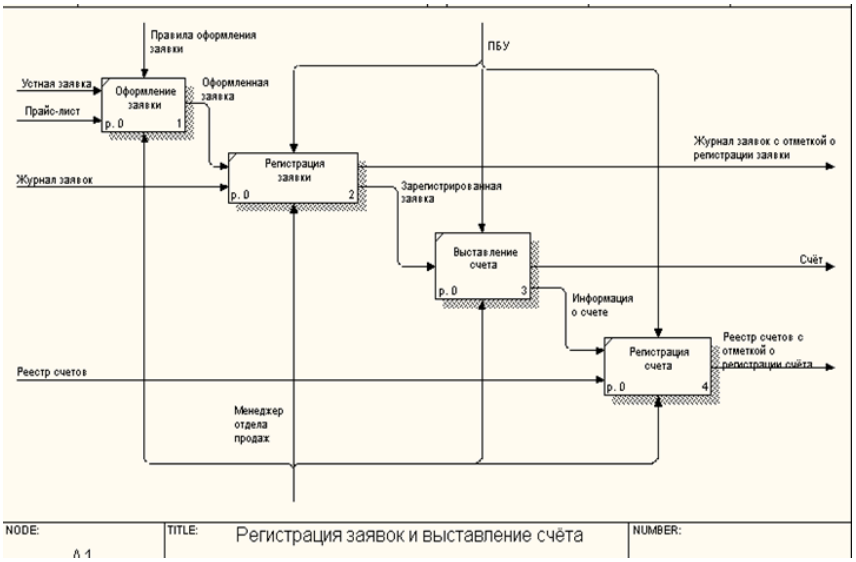

Рис. 4.3 – Первый уровень декомпозиции

Задание:

1. В соответствии с индивидуальным заданием выполните функциональную декомпозицию системы первого уровня с применением подхода IDEF0. Постарайтесь добиться отражения всех высокоуровневых функций системы на первом уровне декомпозиции.
2. Выполните второй уровень декомпозиции для самой сложной функции.
3. Определите списки входных и выходных документов, ресурсов и регулирующих потоков.

Контрольные вопросы

1. Для чего применяется методология IDEF?
2. Что такое «функциональный блок»?
3. Чем отличаются «управление» и «механизмы», входящие в функциональный блок?
4. Как получить список высокоуровневых функций системы с применением диаграмм IDEF0?

Лабораторная работа №5
«МОДЕЛИ ПРЕДСТАВЛЕНИЯ АРХИТЕКТУРЫ, ПРЕДСТАВЛЕНИЕ СЦЕНАРИЕВ»

Цель работы: научиться строить сценарное представление архитектуры.
Форма отчета: продемонстрировать выполненное задание преподавателю

Методические указания:

Теоретические сведения.

После определения видения и контекста, а также проектирования свойств продукта (документы Vision&Scope и Product Features) уже выяснены высокоуровневые функциональные требования и основные нефункциональные требования к системе. После этого производится описание архитектуры в соответствии с какой-либо методикой (4 + 1, Rational Unified Process и др.)

Любые методики представления архитектуры системы предполагают включение её описаний с различных сторон, так называемых архитектурных представлений. Так, в представлении 4 + 1 (модели Кратчена) центральным компонентом являются сценарии (рис. 5.1).

Рис. 5.1 – Модель представления архитектуры «4+1» (модель Кратчена)

Сценарии, как правило, представляются UML диаграммой прецедентов (Use cases).

Задание.

1. Постройте UML диаграмму прецедентов по вашему индивидуальному заданию. Постарайтесь выявить и отобразить связи между действующими лицами и функциями (особое внимание уделите наиболее сложной функции, декомпозицию которой вы выполнили в п.1).
   a. Выделите и опишите всех действующих лиц
   b. Выделите и опишите все прецеденты
   c. Выделите и опишите интерфейсы, если таковые используются
2. Опишите основной и альтернативный сценарий для самого сложного прецедента (см. пп. 2 и 3).
3. Сопоставьте информацию, которая отражена на диаграммах IDEF0, построенных в предыдущей работе и на диаграмме прецедентов. Определите отличия.

Для UML Use case выполните общие описания, используя приведенные ниже таблицы:

Таблица 1. Описание действующих лиц

№ п. п.	Имя	Описание	Abstract (true/false)

Таблица 2. Описание прецедентов

№ п. п.	Имя	Описание	Abstract (true/false)

Контрольные вопросы

1. Как соотносятся блоки диаграмм IDEF0 и прецеденты Use case диаграммы UML?
2. На что из нотации Use case UML похож список ресурсов и механизмов IDEF0? Есть ли различия?
3. Можно ли заменить Use case диаграмму UML диаграммой IDEF0 и наоборот?
4. Чем отличается прецедент от абстрактного прецедента?
5. Когда используются стереотипные зависимости <<include>> и <<extend>>?
6. Где можно использовать отношения обобщения на диаграмме вариантов использования?

Лабораторная работа №6
«МОДЕЛИ ПРЕДСТАВЛЕНИЯ АРХИТЕКТУРЫ, ЛОГИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ»

Цель работы: научиться строить логическое представление архитектуры.
Форма отчета: продемонстрировать выполненное задание преподавателю

Методические указания:

Теоретические сведения.

Логическое представление архитектуры информационной системы представляет концептуальную модель системы и включает подробное описание объектной модели системы и важных архитектурных пакетов. Чтобы описать данное архитектурное представление можно воспользоваться следующими диаграммами унифицированного языка моделирования UML:

- диаграммой пакетов;
- диаграммой моделей;
- диаграммой классов.

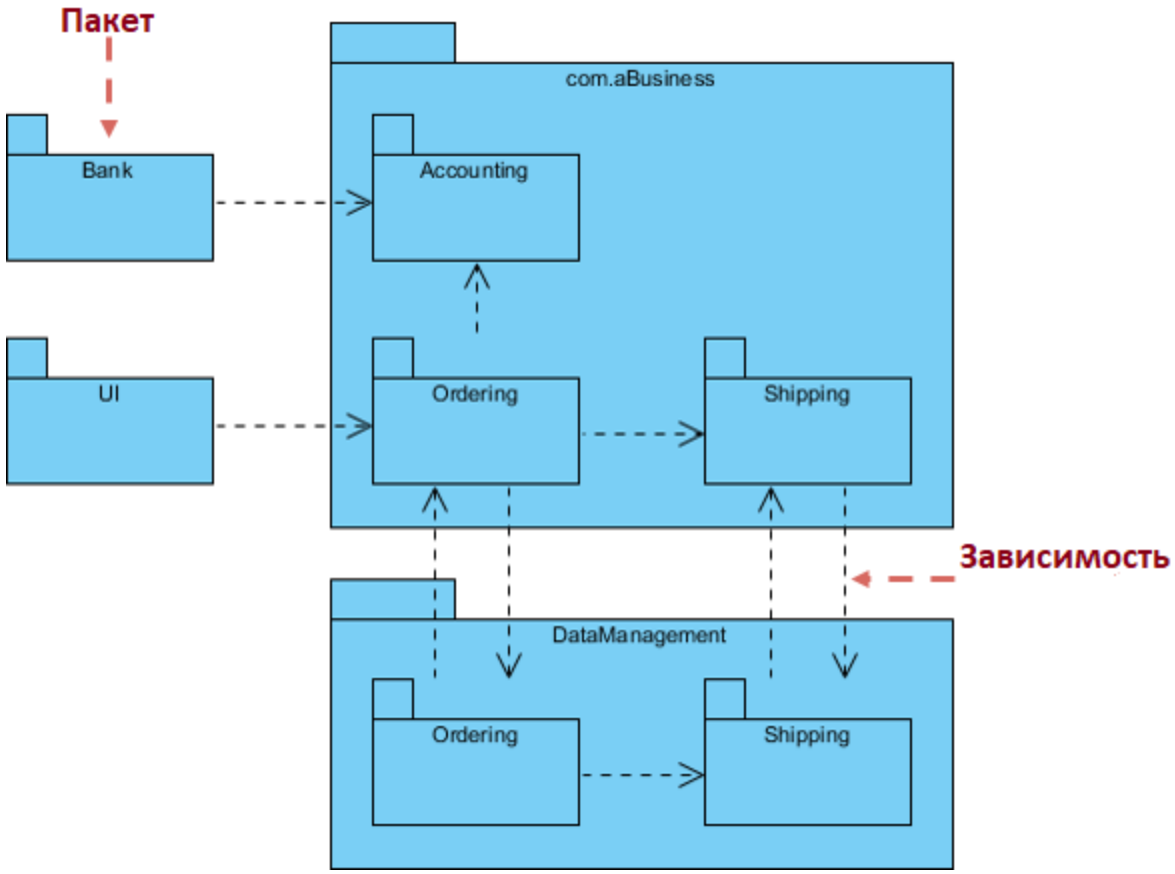

Рис. 6.1 – Диаграмма пакетов UML

В случае, если требуется показать дополнительные архитектурные свойства: звенья, слои, группы пакетов и зависимости между ними, то можно использовать диаграмму моделей UML. Пример показан на рис. 6.2.

Как видно из показанного примера, диаграмма моделей отражает не только пакеты, но и

архитектурные слои системы, а также взаимосвязи между ними. Так, видны 3 слоя: представления, бизнес-логики, данных и их пакеты.

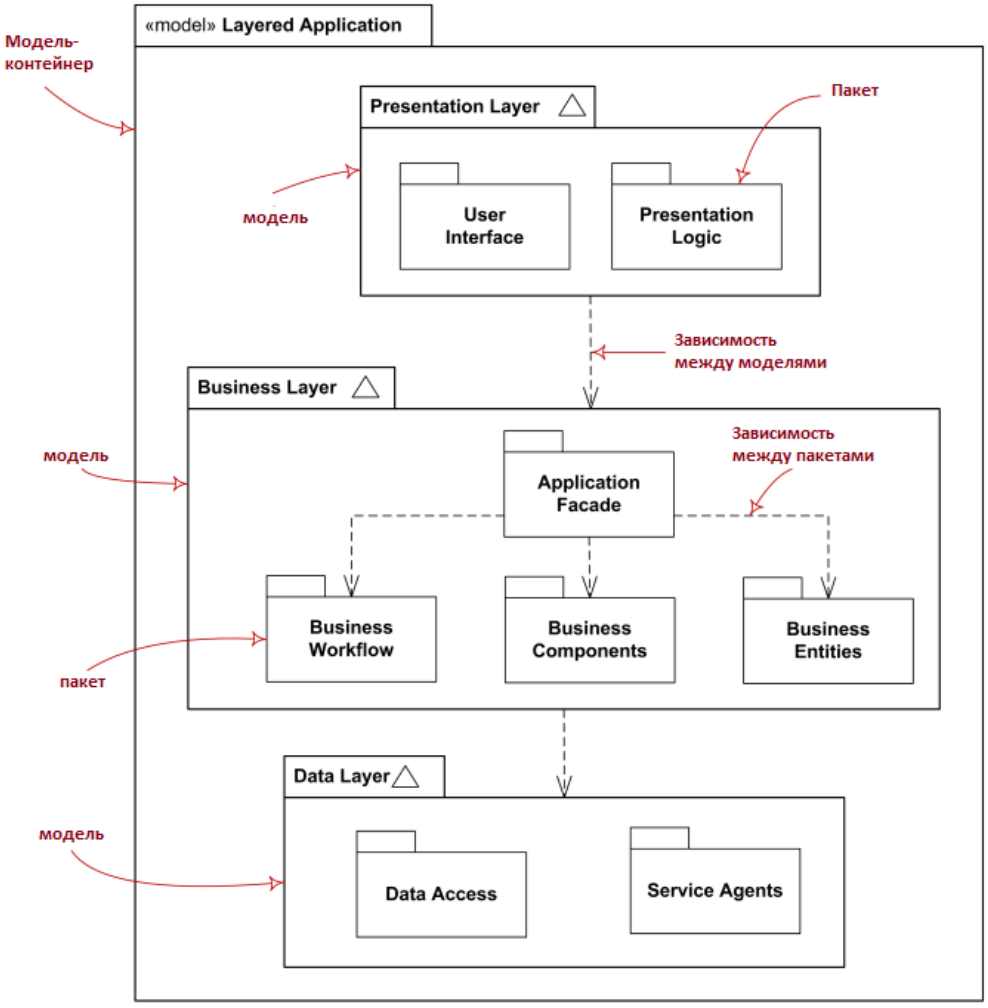

Рис. 6.2 – Диаграмма моделей UML

Дальнейшая детализация производится с применением диаграммы классов. Данная диаграмма подробно изучается на курсах объектно-ориентированного программирования и проектирования [1].

Задание.

1. Определите основные архитектурно значимые пакеты системы.
2. Опишите взаимодействие частей системы с помощью диаграммы моделей UML. Укажите основные слои проектируемой системы, взаимозависимости между моделями и зависимости между пакетами.
3. Для пакетов/моделей, которые так или иначе отражают логику обработки данных в разрабатываемой системе, постройте диаграмму классов, которая отражает основные абстракции системы (интерфейсы, абстрактные классы и сущности). В случае высокой загруженности диаграммы рекомендуется разделить её на две:
   a. Диаграмму классов с основными сущностями
   b. Диаграмму классов с основными абстракциями логики

Контрольные вопросы

1. Что такое «пакет» в терминах проектирования информационных систем?
2. Что отражает диаграмма пакетов UML?
3. В чём разница между диаграммой пакетов и диаграммой моделей UML? Являются ли они взаимозаменяемыми?
4. Как оценить степень зависимости между пакетами и моделями?
5. Что отображают на диаграмме классов в документах уровня архитектуры?
6. Какую информацию доносит до проектировщиков и разработчиков логическое представление архитектуры?

Лабораторная работа №7
«МОДЕЛИ ПРЕДСТАВЛЕНИЯ АРХИТЕКТУРЫ, ПРЕДСТАВЛЕНИЕ ПРОЦЕССА»

Цель: научиться строить представление процесса архитектуры, описывать потоки данных и их синхронную и асинхронную обработку.
Форма отчета: продемонстрировать выполненное задание преподавателю

Методические указания:

Теоретические сведения.

Представление процесса – это архитектурное представление, которое отражает поведение, параллелизм, информационные потоки и некоторые аспекты нефункциональных требований. Одним из вариантов представления является UML диаграмма состояний. Эта диаграмма отображает набор состояний и переходы между ними, включая условия (рис. 7.1).

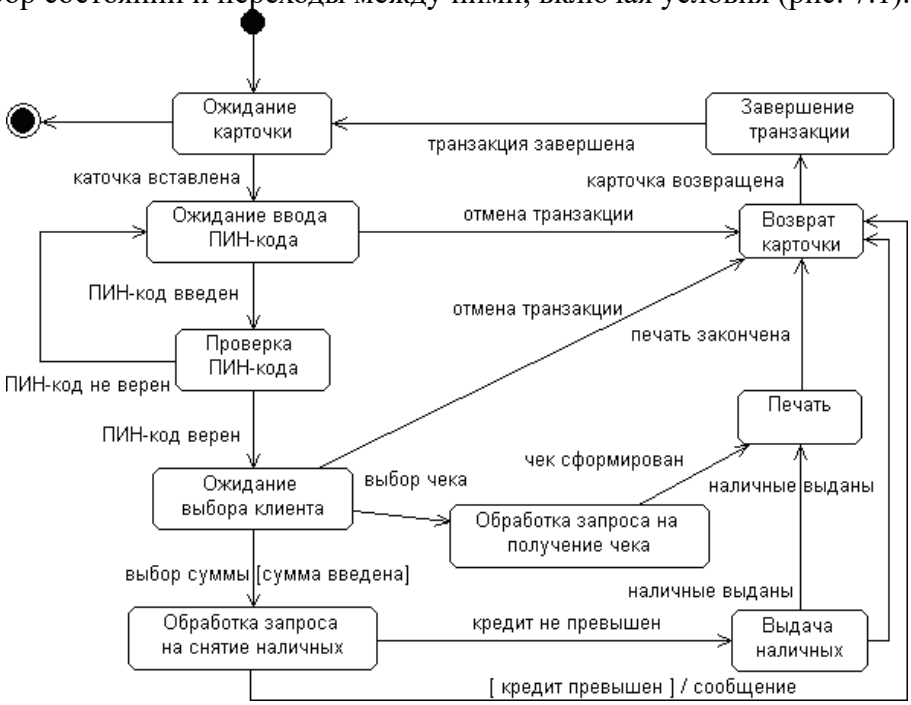

Рис. 7.1 – Пример диаграммы состояний UML

Диаграмма состояний состоит из состояний (начального, конечного и промежуточных) и переходов. В языке UML под состоянием понимается абстрактный метакласс, используемый для

моделирования отдельной ситуации, в течение которой имеет место выполнение некоторого условия [2, 3].

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий. В этом состоянии находится объект по умолчанию в начальный момент времени.

Конечное (финальное) состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий. В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени.

Переход представляет – это связь между двумя последовательными состояниями, указывающая на факт смены одного состояния другим.

Другими вариантами описания процесса являются: диаграмм информационных потоков (DFD), диаграмма действий (UML Activity) и диаграмма последовательностей (UML Sequence).

Нефункциональными аспектами, имеющими отношение к процессу, являются целостность, надёжность, доступность и параллелизм. Как правило, их описывают в текстовом виде, так как это проще и удобнее.

Задание.

1. Выберите основной сценарий бизнес-логики из Use Case, построенных ранее.
2. Опишите этот сценарий UML диаграммой состояний.
3. Постройте диаграмму информационных потоков для пакета, содержащего выбранный Use Case.
4. Опишите нефункциональные требования, с точки зрения процесса.

Контрольные вопросы

1. Какие подходы и инструментальные средства пригодны для описания представления процесса?
2. Чем отличаются диаграммы, описывающие систему с точки зрения процесса?
3. В чём отличие информационных потоков от переходов между состояниями?
4. Почему нефункциональные аспекты процесса описываются отдельно от диаграмм?

Лабораторная работа №8
«МОДЕЛИ ПРЕДСТАВЛЕНИЯ АРХИТЕКТУРЫ, ФИЗИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ»

Цель работы: научиться строить физическое представление архитектуры, с учётом атрибутов качества и требований к развёртыванию системы.

Форма отчета: продемонстрировать выполненное задание преподавателю

Методические указания:

Теоретические сведения.

Наибольший интерес для системных администраторов, команд поддержки и, вообще, специалистов из сферы DevOps представляет распределение программных модулей по физическим узлам сетевой инфраструктуры и виртуальным машинам. Именно для этого используется физическое представление архитектуры. Оно включает в себя, как правило:

- диаграмму развёртывания UML (UML deployment diagram);
- описание двух атрибутов качества: ёмкости системы и управления конфигурацией (управляемости).

Основными целями, преследуемыми при разработке диаграммы развертывания,

являются [4]:
- распределение компонентов системы по ее физическим узлам;
- отображение физических связей между узлами системы на этапе исполнения;
- выявление узких мест системы и реконфигурация ее топологии для достижения требуемой производительности.

Элементами диаграммы являются узлы, компоненты системы и связи между ними.

Узел – это некоторый физический компонент системы. В его качестве может выступать сервер, рабочая станция, датчик или иное устройство, содержащее программное обеспечение или непосредственно взаимодействующее с ним.

Связь – это физический канал соединения, который обеспечивает взаимодействие между узлами, или зависимость между компонентами системы.

Пример диаграммы развёртывания с учётом спецификаций приведен ниже, на рис. 8.1.

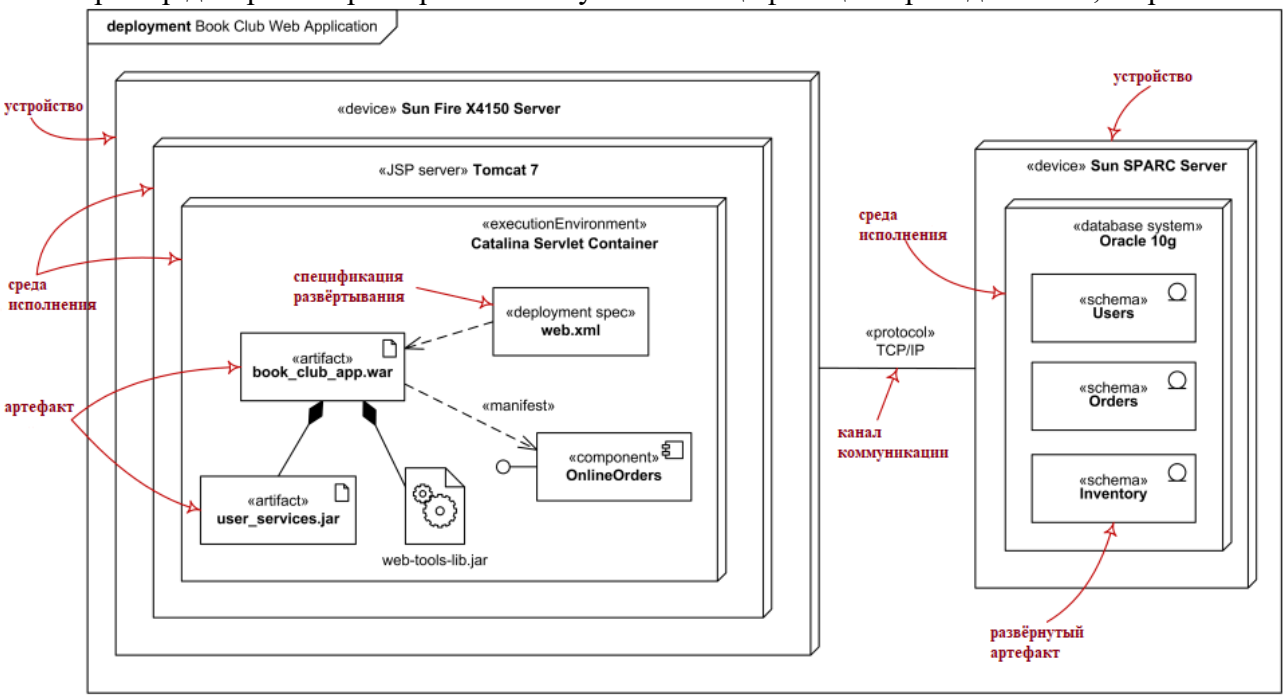

Рис. 8.1 – Диаграмма развёртывания UML

Задание.

1. Создайте физическое представление пакета бизнес-логики вашей проектируемой информационной системы.

a. Опишите сетевую инфраструктуру, в условиях которой должна работать система и создайте диаграмму развёртывания

b. Опишите требования к ёмкости системы

c. Опишите требования к управляемости системы

Контрольные вопросы

1. Что такое ёмкость и управляемость системы? Как задать требования к этим величинам количественно?
2. Для чего предназначена диаграмма развёртывания?
3. Какие элементы содержит диаграмма развёртывания?
4. Кто наиболее заинтересован в документации по физическому представлению архитектуры системы?

ЛИТЕРАТУРА

1. Мартин Фаулер, UML. Основы. Краткое руководство по стандартному языку объектного моделирования // СПб.: Символ плюс. – 2018. – 192с.
2. Крээг Ларман, Применение UML 2.0 и шаблонов проектирования // М.: Вильямс. – 2019. – 736с.
3. Пашкевич А. П., Чумаков О. А., Современные технологии программирования // Минск: 2007. – URL: https://studizba.com/lectures/10-informatika-i-programmirovanie/368-sovremennye-tehnologii-programmirovaniya/4989-10-diagrammy-sostoyaniy.html
4. Хассан Гома, UML. Проектирование систем реального времени, параллельных и распределенных приложений // М.: ДМК Пресс. – 2016. – 700с.