Системный Анализ. Предметная область

Михаил Кумсков

Модели на UML

Михаил Кумсков

Системный Анализ. Предметная область. Модели на UML

«Издательские решения»

Кумсков М.

Системный Анализ. Предметная область. Модели на UML /
М. Кумсков — «Издательские решения»,

ISBN 978-5-00-509385-1

Конспект лекции по определению модели предметной области на конкретном примере. Используется подход, существенно отличающийся от известного ER-моделирования. Модель имеет «визуальный характер» и изображается в нотации Unified Modeling Language (UML), которая «широко известна в узких кругах» аналитиков, архитекторов, разработчиков и программистов. Описаны паттерны, применяемые для преобразования диаграмм классов на UML и приведены примеры их практического использования.

ISBN 978-5-00-509385-1

Содержание

Системный Анализ. Предметная область Модели на UML

Михаил Кумсков

ISBN 978-5-0050-9385-1

Создано в интеллектуальной издательской системе Ridero

Книга представляет собой краткий конспект лекций по определению модели предметной области на конкретном примере. Используется объектно-ориентированный подход, существенно отличающийся от известного моделирования «сущность – связь», или ER-моделирования. Модель имеет визуальный характер и изображается в нотации Unified Modeling Language (UML), которая широко известна среди аналитиков, архитекторов, разработчиков и программистов. Описаны паттерны, применяемые для преобразования диаграмм классов на UML, и приведены примеры их практического использования. Изложение ведется согласно методологии IBM RUP.

Материал будет полезен студентам и аспирантам, участникам проектов по разработке информационных систем, а также слушателям курсов по выявлению требований к ИС и по проектированию архитектуры ИС.

Введение

Модель предметной области служит разным целям:

1) помогает определить логическую структуру БД информационной системы;

2) является основой для составления «расширенного» словаря проекта;

3) помогает найти все сценарии (при выявлении функциональных требований в специальной форме – в виде сценариев использования (Use Cases));

4) позволяет не пропустить «вспомогательные» сценарии, которые могут быть не упомянуты в постановке задачи, полученной от заказчика ИС.

Важной особенностью модели предметной области является ее ***независимость*** от используемых ИС и баз данных. Это совокупность записей в организации, которые следует вести, чтобы руководство этой организации смогло определить, все ли в порядке. Например, такие записи велись много лет назад задолго до появления компьютеров. Чтобы не было разночтений, определим некоторые используемые далее термины. Их понимание важно, но они могут отличаться от общеизвестных своей специфической направленностью.

• *Модель* – это «упрощение реальности» в интересах заинтересованных лиц.

• *База данных* – набор картотек, взаимосвязанных друг с другом и ведущихся на компьютере.

• *Сервер* – процесс, предоставляющий целостный доступ к общему ресурсу.

• *Сервер БД*, или СУБД (система управления БД), предоставляет целостный доступ к базе данных как общему ресурсу.

• *Картотека* – набор карточек с «одинаковой структурой», представляется в модели *классом*.

• *Карточка* – запись в картотеке, представляется в модели *объектом*. Все карточки одной и той же картотеки имеют одинаковую структуру: одинаковый набор полей карточки (или атрибутов) и одинаковый набор связей с карточками других картотек.

• *Класс* – это описание набора «одинаковых» объектов, т. е. объектов, имеющих одинаковый *набор атрибутов*, одинаковый набор *операций* и одинаковый набор указателей на другие объекты. Картотеки представимы классами.

• *Объект* – это экземпляр класса, т. е. запись, или «карточка», в соответствующей картотеке.

• *Атрибут* – поименованное свойство объекта.

• *Операция* – сервис, который может быть запрошен у объекта. Метод или функция, инкапсулированная в объект.

Для проведения визуального моделирования будем использовать специальные программные инструменты, называемые CASE-средствами (Computer Assist Software Engineering)[1]. Тогда будет возможно проведение «генерации кода» по модели («прямое проектирование», или forward engineeging) и обратное проектирование (reverse engineering) – восстановление модели по программному коду или по существующей БД.

Книга состоит из двух разделов. В первом описан пошаговый процесс выявления элементов модели и построения набора диаграмм классов UML как модели предметной области. Второй раздел вводит понятие процесса выявления требований к ИС в специальной форме «сценариев использования» (UC – Use Cases) и разъясняет, как использовать модель предметной области для выявления сценариев использования.

[1] Если пользоваться графическими редакторами, например, MS Visio, то возникнут проблемы синхронизации визуальных элементов при «разрастании» (масштабировании) модели.

Модель предметной области отличается от моделирования «сущность – связь», или ER-моделирования (Entity-Relationship), методически и содержательно. ER-модель служит для описания логической структуры БД и имеет собственную визуальную нотацию. С технической точки зрения в сущностях визуальной ER-модели следует явно «прописывать» так называемые первичные и внешние ключи реляционной модели данных. Эти отличия кратко излагаются в приложении.

Ранее представленная методика была описана в методическом пособии Кумсков М. И. Базы данных и процессы их создания. Введение. М.: Мехмат МГУ, 2004.

Благодарности

Появлению этой книги способствовали общение и совместная работа с многими профессионалами своего дела. Пользуюсь случаем и выражаю свою признательность:

Бахвалову Николаю Сергеевичу,

Позину Борису Ароновичу,

Сорокину Александру Викторовичу,

Ивановой Елене Владимировне,

Авдошину Сергею Михайловичу.

Представленные ниже методики и материалы прошли апробацию на курсах и тренингах, читаемых на мехмате МГУ, в департаменте программной инженерии факультета компьютерных наук ВШЭ, в учебном центре «Люксофт», при проведении мастер-классов и выступлений на конференциях. Видеозапись некоторых из них можно найти по следующим ссылкам:

1. Lviv IT Arena 2015: Mikhail Kumskov, Best Practices of Project Execution According to IBM Rational. URL: https://www.youtube.com/watch?time_continue=2&v=p5NzDKDzLOY

2. Value Management and Business Analysis (VM&BA). Mikhail Kumskov. Workshop. URL: https://www.youtube.com/watch?v=6FEUlwXrfgQ

3. Analyst Day 2014. Синергия UML: Модель предметной области, Бизнес-системы, Информационные системы: переход шаг за шагом. URL: https://www.youtube.com/watch?time_continue=5&v=Vl6SFx0rzqw

4. Летний аналитический фестиваль 2013 (ЛАФ-2013) «Системный анализ ИС и бизнес-системы – связь, сходства и различия». URL: https://www.youtube.com/watch?time_continue=3&v=4LtIQVj3juw

5. Конференции REQ Labs 2011. Процессы и люди. URL: https://www.youtube.com/watch?v=cz5IBkf5E20

Раздел 1

Построение визуальной модели предметной области

Модель – это «упрощение реальности» в интересах заинтересованных лиц. Такое определение относится и к нашему моделированию. Здесь главным заинтересованным лицом является инвестор или топ-менеджер организации. Есть и другие заинтересованные лица – аналитики, архитекторы, разработчики информационной системы (ИС), и поэтому одной модели, как правило, недостаточно. Нужны разные «упрощения» для разных читателей модели[2].

Первым шагом процесса моделирования является определение *целей* моделирования. Будем содержательно разбирать процесс построения на примере ИС, учитывающей расход продуктов в кафе и ресторанах организации, которую назовем «Комбинат питания». Текст с описанием задачи, полученный от владельца комбината, приведен в начале приложения 1.

Общий взгляд на процесс, состоящий из семи шагов, можно представить следующим списком задач, выполняемых в ходе моделирования:

- Шаг №0. Определяем цели построения модели.
- Шаг №1. Определяем события-картотеки, подлежащие учету на предприятии.
- Шаг №2. Определяем справочники-картотеки, подлежащие учету.
- Шаг №3. Для события определяем картотеки, связанные с ним (для каждого события).
- Шаг №4. Для справочника определяем картотеки, связанные с ним (для каждого справочника).
- Шаг №5. Отображаем (визуально) картотеки, связанные с ней на диаграмме классов UML.
- Шаг №6. Применяем паттерны преобразования отношений на диаграммах классов UML.

Шаг №0. Определяем цели построения модели

Цель построения модели в задаче «Комбинат питания» была определена в постановке задачи.

Это *учет заказов гостей, движения продуктов и денег за них в пунктах питания* – кафе и ресторанах. Теперь мы не будем учитывать и вводить в модель те детали, которые не относятся к заявленной цели. Например, не учитываем события «бронирование столика в кафе».

Далее следует определить те ***события*** («бизнес-транзакции»), которые подлежат учету, согласно заданным целям. Для таких событий на предприятии будут вестись учетные записи, или – в нашей терминологии – ***картотеки*** (гроссбухи, если учет бумажный).

Шаг №1. Определяем события, подлежащие учету

Для нашего примера мы выявляем бизнес-события по «движению продуктов питания и денег за них». Очевидно, что такими событиями будут:

1. «**Заказ**» гостя.
2. «**Оплата заказа**».
3. «**Покупка продуктов**» в кафе.

---

[2] «Сложность – это простота, изложенная подробно». Такое определение перекликается с понятием моделирования как «упрощения „сложной“ реальности».

Для каждого события определяется картотека – при возникновении события в этой картотеке должна быть создана новая учетная запись (карточка).

Для выявления других событий будем использовать *паттерны*[3]. Первым паттерном является введение «расхода» учетных сущностей – продуктов и блюд – через «*брак*» или «некачественную сущность», подлежащую списанию. По этому паттерну («Списание брака») вводим два новых события:

4. «**Списание бракованных продуктов**» (по паттерну).
5. «**Списание бракованных блюд**» (по паттерну).

Следующий паттерн называется «**Инвентаризация**»: если на предприятии есть учетная система (информационная или на бумаге), то периодически следует выявлять реальный состав предметов учета на складе. Полученные в результате инвентаризации списки предметов (в нашем случае – продуктов питания) следует сравнить со списками, полученными (как отчеты) из учетной системы. Вводим в наш список событие:

6. «**Инвентаризация**» склада кафе (по паттерну).

Итак, пометка «по паттерну» означает, что данное событие может быть не указано в постановке задачи, но мы применяем паттерн (стандартное решение стандартной часто встречающейся задачи). Паттерн «**Списание брака**» означает учет (некачественных изделий) – в нашем случае – продуктов и блюд. Паттерн «**Инвентаризация**» используется, если имеется некоторая учетная система и требуется периодически сравнивать фактический состав продуктов на складе со «списочным составом», полученным из учетной системы. Событие «инвентаризация» – это проведение учета фактического состава продуктов на складе кафе ресторана на данную календарную дату. Проведение сравнения двух списков («Инвентаризация» – «Отчет системы») позволяет выявить «недостачу».

Третьим паттерном является паттерн «**Прайс-лист**», который в комбинате питания называется «меню». Цены блюд в каждом кафе могут различаться. Для этого ведется меню как прайс-лист блюд на данную дату работы кафе. Поскольку для цен определена дата (или интервал дат) их «действия», то эту сущность мы отнесем к бизнес- событиям.

7. «**Меню**» кафе (по паттерну).

В результате мы выявили семь событий, подлежащих учету, – три непосредственно из постановки задачи и еще четыре на основе применения паттернов.

Шаг №2. Определяем справочники, подлежащие учету

Кроме картотек по событиям, на предприятии ведутся «учетные списки», которые можно назвать ***справочниками.*** Справочники более «стабильны» – изменения в них вносятся значительно реже и они «не зависят» от даты. Например, в комбинате питания следует вести список всех «пунктов питания» (ресторанов, кафе, закусочных). Это же относится и к списку сотрудников комбината. Такие справочники также являются картотеками, и они также подлежат учету в нашей модели как классы:

1. «**Пункт питания**» (ПП).
2. «**Сотрудник**».
3. «**Блюдо**» (рецепты).

---

[3] *Паттерн* — стандартный шаблон решения стандартной задачи, который хорошо себя зарекомендовал в прошлых проектах.

4. «**Продукты**»[4] (включая напитки).

Следует понимать, что на первых этапах могут быть выявлены не все события и справочники. Новые картотеки будут появляться далее в ходе итерационного построения модели, при применении паттернов и при получения обратной связи от заинтересованных лиц.

Шаг №3. Для события определяем картотеки, связанные с ним (для каждого события)

Карточки могут быть связаны с карточками в других картотеках. Это обусловлено принципом учетных систем: *«Информацию об объекте следует вводить только один раз и использовать затем много раз»*. Нам в модели нужно указать связи картотек друг с другом. Для этого рассмотрим все события по очереди и для каждого определим связанные с ним картотеки и справочники согласно предметной области.

1. «**Заказ**»

При определении связей «Заказа» рассматриваем все выявленные на шагах 1 и 2 классы-картотеки и смотрим, содержит ли другая картотека важную информацию для данного события. Важная информация по заказу: в каком кафе был сделан данный заказ («Кафе»), кто проводил обслуживание гостя («Сотрудник»), какие блюда составляли заказ («Блюдо»), какие цены были на блюда («Меню»), был ли заказ оплачен («Оплата»)? В результате имеем следующие связи класса «Заказа»:

Номер	*Событие-картотека*	*Связанные картотеки*
1	«Заказ»	«Кафе» «Оплата заказа» «Сотрудник» «Меню» «Блюдо»

2. «**Оплата заказа**»

«Оплата заказа» связана только с заказом и без него не может существовать. У нас эта связь уже найдена.

[4] Включая напитки. Названия справочников приведены в единственном числе, поскольку название картотеки – это название класса.

2	«Оплата Заказа»	«Заказ»

3. «**Покупка продуктов**» в кафе

Важная информация о закупке продуктов, которая содержится в других списках: в каком кафе была закупка, каким сотрудником осуществлена, какие продукты были закуплены.

3	«Покупка продуктов»	«Кафе» «Сотрудник» «Продукты»

Результаты выявления связей других событий представим в виде таблицы 1.1. «Связь картотеки-события с другими картотеками».

Таблица 1.1. ***Связь картотеки-события с другими картотеками***.

Номер	*Событие-картотека*	*Связанные картотеки*
4	«Списание брак. продуктов»	«Кафе» «Сотрудник» «Продукты»
5	«Списание брак. блюд»	«Кафе» «Сотрудник» «Блюдо»
6	«Инвентаризация»	«Кафе» «Сотрудник» «Продукты»
7	«Меню»	«Кафе» «Блюдо»

По аналогии найдем связи справочников друг с другом.

Шаг №4. Для справочника определяем картотеки, связанные с ним (для каждого справочника)

Результаты выявления связей оформим в виде таблицы 1.2.

Таблица 1.2. ***Связи справочника***

Номер	*Справочник*	*Связанные картотеки*
8	«Кафе»	«Сотрудники» + связи с событиями
9	«Сотрудники»	«Кафе» + связи с событиями
10	«Блюдо»	«Продукты» + связи с событиями
11	«Продукты»	«Блюдо» + связи с событиями

Для каждой картотеки создаем соответствующий класс в репозитории выбранного для моделирования CASE-инструмента. Всего следует создать 11 классов: 7 классов для событий и 4 класса для справочников. Соответствующие диаграммы классов изображены на рисунках 1.1 и 1.2.

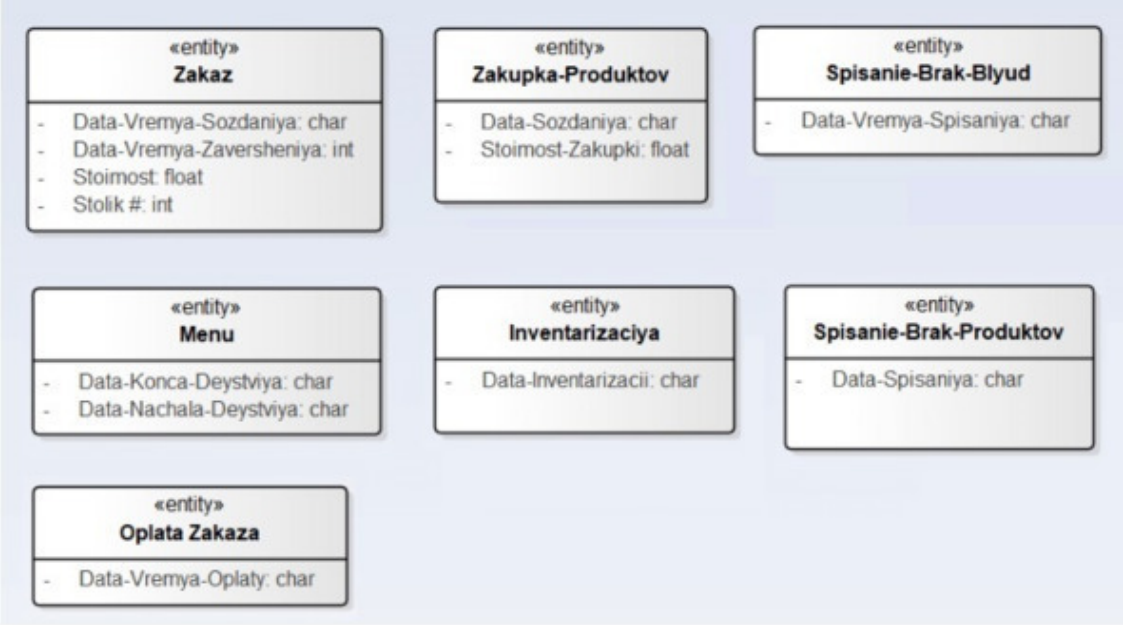

Рис. 1.1. Диаграмма классов UML для картотек событий. Показаны атрибуты классов

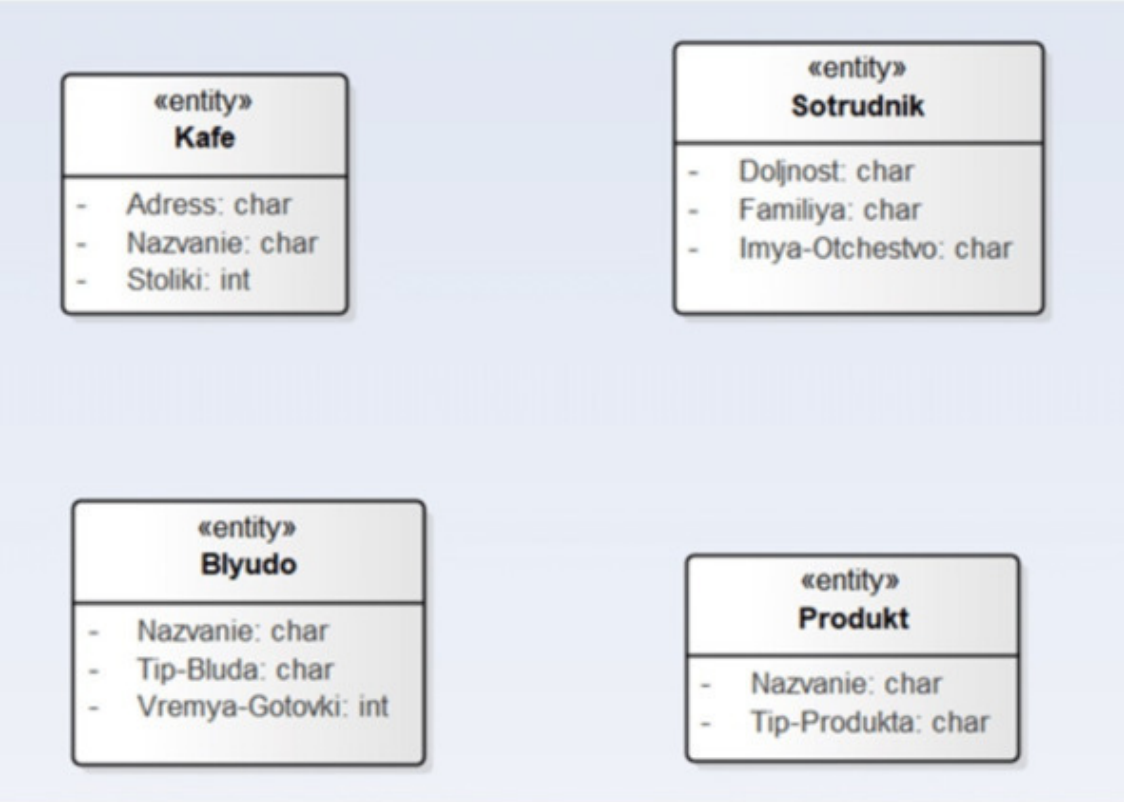

рис. 1.2. диаграмма классов uml для справочников картотек. показаны атрибуты классов

Шаг №5. Отображаем (визуально) картотеки, связанные с ней на диаграмме классов UML

Создаем новую диаграмму классов[5] и помещаем в ее центр класса «Заказ гостя». Согласно списку связей указываем связанные с ним классы-картотеки. После расстановки связей между классами, которые на UML называются **ассоциациями**, для каждого *конца ассоциации* расставляем «**множественность**».

Множественность определяется для *экземпляров*, т. е. в нашем случае для карточек, и указывает, сколько карточек связано с данной карточкой:

Данный «**Заказ**» связан:

• с одним экземпляром «**Кафе**» (значок «1» – ровно один);

• с одним экземпляром класса «**Сотрудник**» (значок «1» – ровно один) (т. е. с официантом, проводящим обслуживание);

• со многими экземплярами «**Блюд**» (значок «*» – ноль или много);

• с ноль или одной «**Оплатой заказа**» (значок «0..1»);

• с одним «**Меню**» (значок «1» – ровно один).

Теперь рассмотрим другую сторону (роль) ассоциаций:

• экземпляр «**Кафе**» связан со многими «**Заказами**» (значок «0..*» – ноль или много);

• экземпляр «**Сотрудника**» связан со многими «**Заказами**» (значок «0..*»);

• экземпляр «**Блюда»** связан со многими «**Заказами**» (значок «0..*»);

• «**Оплата заказа**» связана ровно с одним «**Заказом**» (значок «1» – ровно один);

[5] Диаграмма классов UML создается в выбранном для моделирования CASE-инструменте.

• экземпляр «**Меню**» связан со многими «**Заказами**» (значок «0..*»).

Теперь указываем «поля карточек» в картотеках как *атрибуты классов*, существенные для учета согласно целям моделирования.

Атрибуты «**Заказа клиента**»: «**дата-время начала обслуживания»** и «**дата-время завершения обслуживания**» (гости ушли, освободив столик) позволяют учитывать время, когда столик («**номер столика**» указан в атрибуте) был занят на время данного обслуживания. Также необходимо учитывать общую «**стоимость**», вычисляемую по «Меню» и количеству заказанных блюд.

• Атрибуты «**Сотрудника**» – фамилия, имя, отчество, должность, оклад (опционально).

• Атрибуты «**Кафе**» – название, адрес, число столиков.

• Атрибуты «**Оплаты заказа**» – дата-время оплаты (стоимость не указываем, т. к. уже есть в самом «Заказе»);

• Атрибуты «**Блюда**» – название, тип блюда, время приготовления.

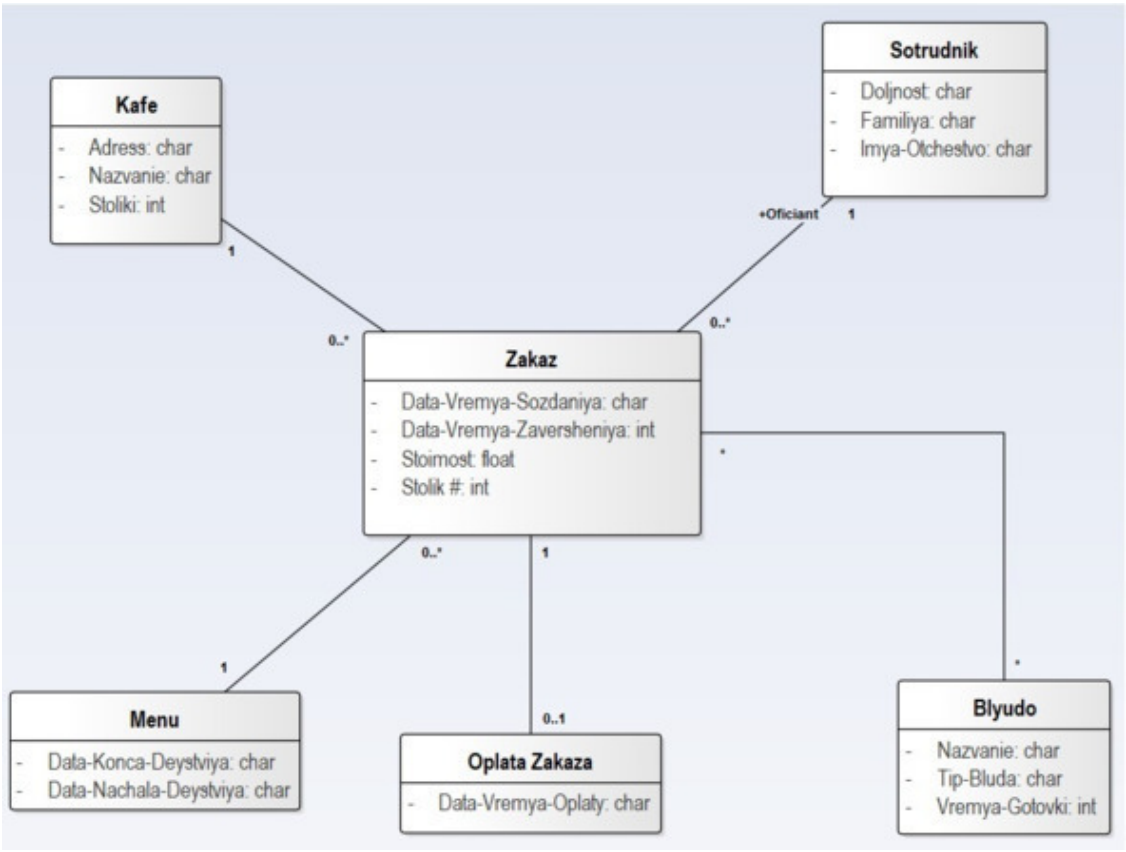

Рис. 1.3. Диаграмма классов UML для картотеки «Заказ» гостя в кафе. Показаны атрибуты классов

Для оставшихся событий «Покупка продуктов», «Списание бракованных продуктов», «Списание бракованных блюд», «Инвентаризация» и «Меню» соответствующие диаграммы классов UML приведены ниже – на рисунках 1.4—1.8.

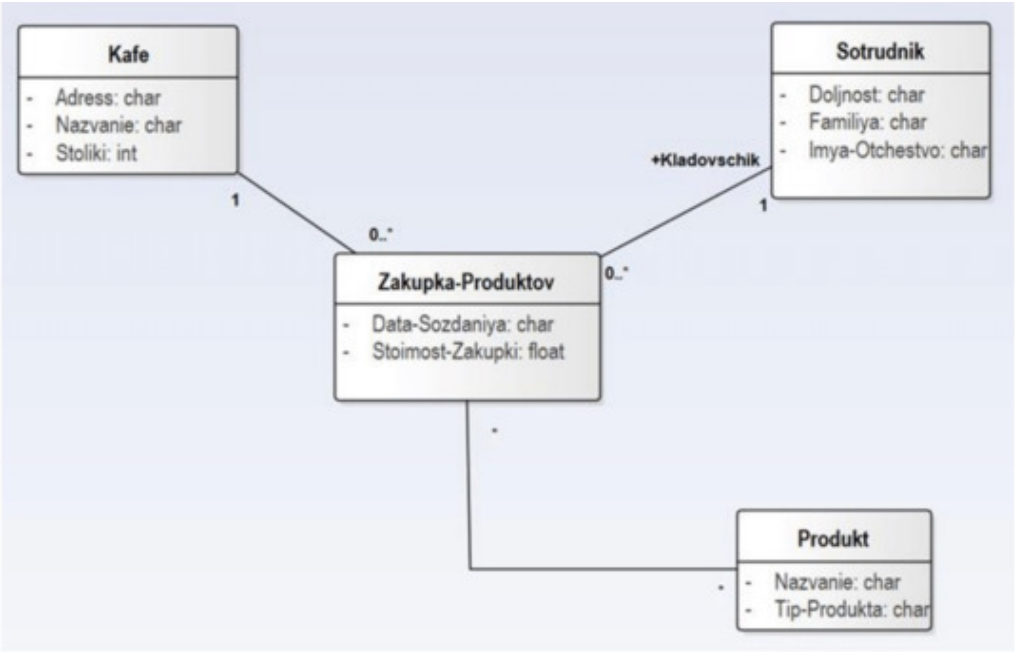

Рис. 1.4. Диаграмма классов UML для картотеки «Закупка продуктов» в кафе. Показаны атрибуты классов участников

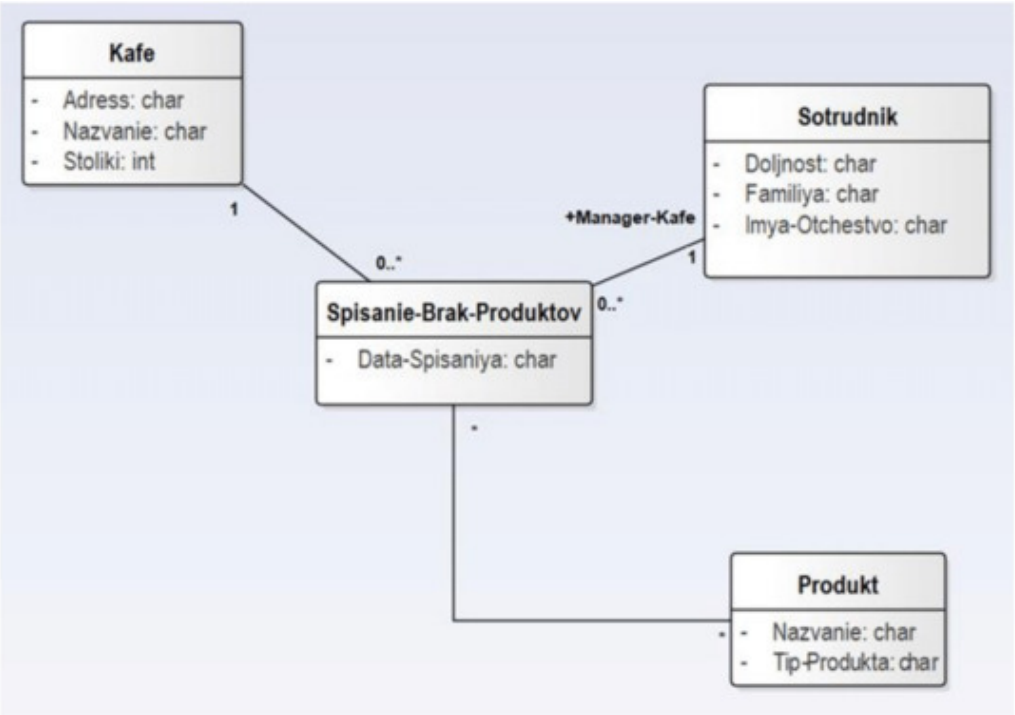

Рис. 1.5. Диаграмма классов UML для картотеки «Списание бракованных продуктов» в кафе

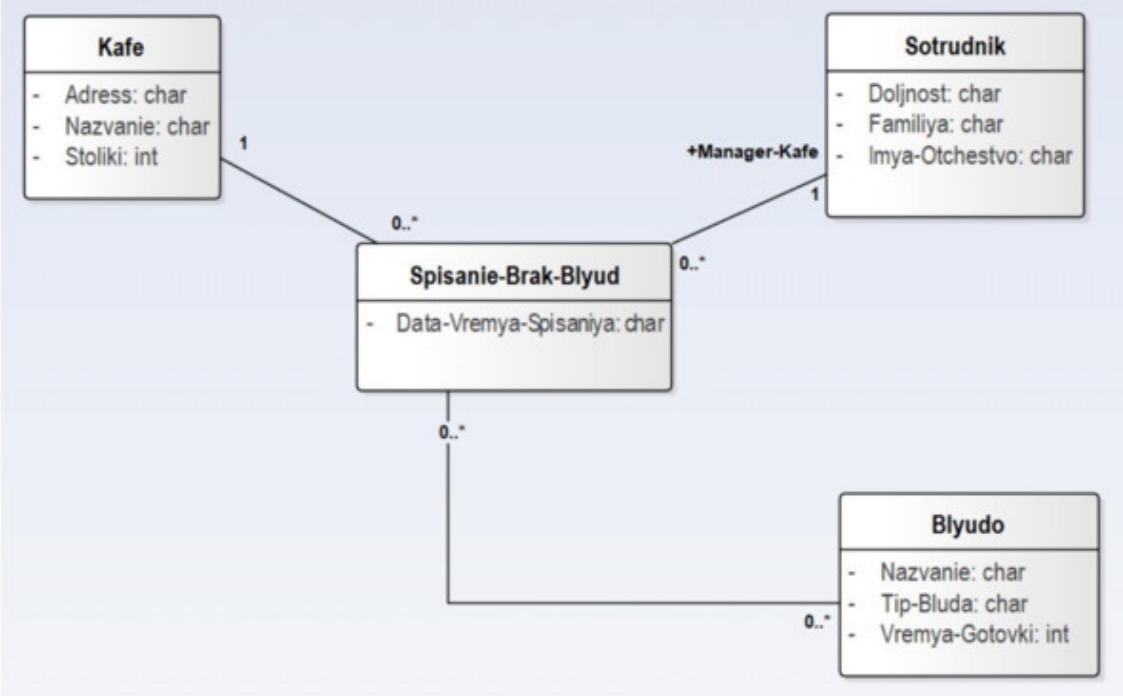

Рис. 1.6. Диаграмма классов UML для картотеки «Списание бракованных блюд» в кафе

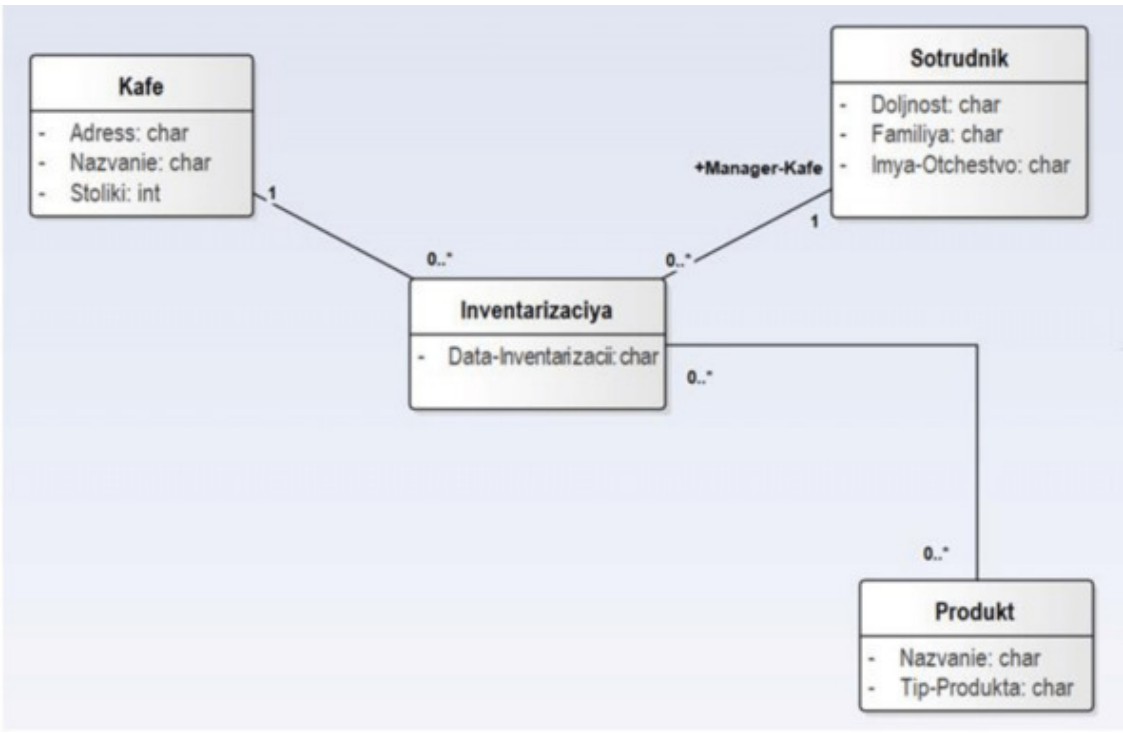

Рис. 1.7. Диаграмма классов UML для картотеки «Инвентаризация» в кафе

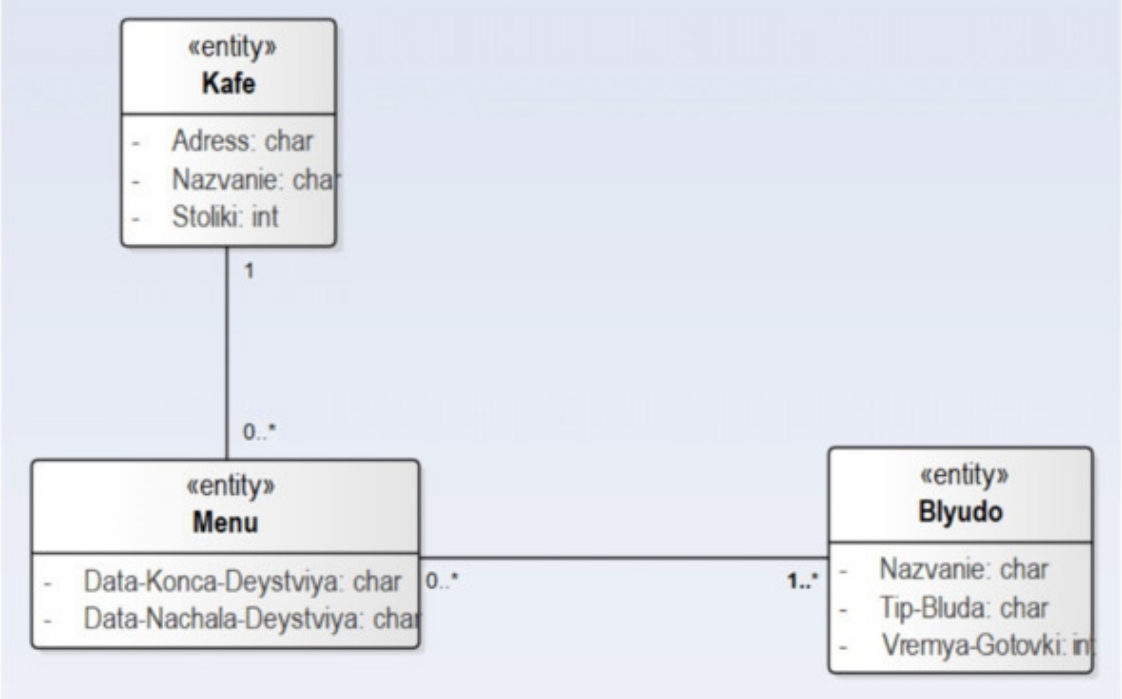

Рис. 1.8. Диаграмма классов UML для картотеки «Меню»

Теперь, продолжая шаг 5, связываем справочники друг с другом, последовательно создавая диаграммы классов для картотек «Пункт питания», «Сотрудник», «Блюдо» и «Продукты».

Рис. 1.9. Диаграмма классов UML для справочников «Кафе» и «Сотрудник»

Рис. 1.10. Диаграмма классов UML для справочников «Блюдо» и «Продукты»

Шаг №6. Применяем паттерны на диаграммах-«ромашках»

Паттерн «Объект-список»

Ассоциации с множественностью «Много ко многим» могут иметь собственные свойства. Например, для связи «Заказ» – «Блюдо» (рис. 1.3) таким свойством является ***число заказанных экземпляров*** блюда. Например, если заказано два мороженых, то двойка – это не атрибут «Заказа» и не атрибут «Блюда» (рецепта), это атрибут ассоциации «много ко многим».

Согласно паттерну «Объект список» удаляем из диаграммы ассоциацию между «Заказом» и «Блюдом» и вместо нее вводим новый класс «Строка списка». Эту строку следует ввести для каждого блюда, вошедшего в «Заказ». Результат применения паттерна «Объект-список» показан на рис. 1.3.

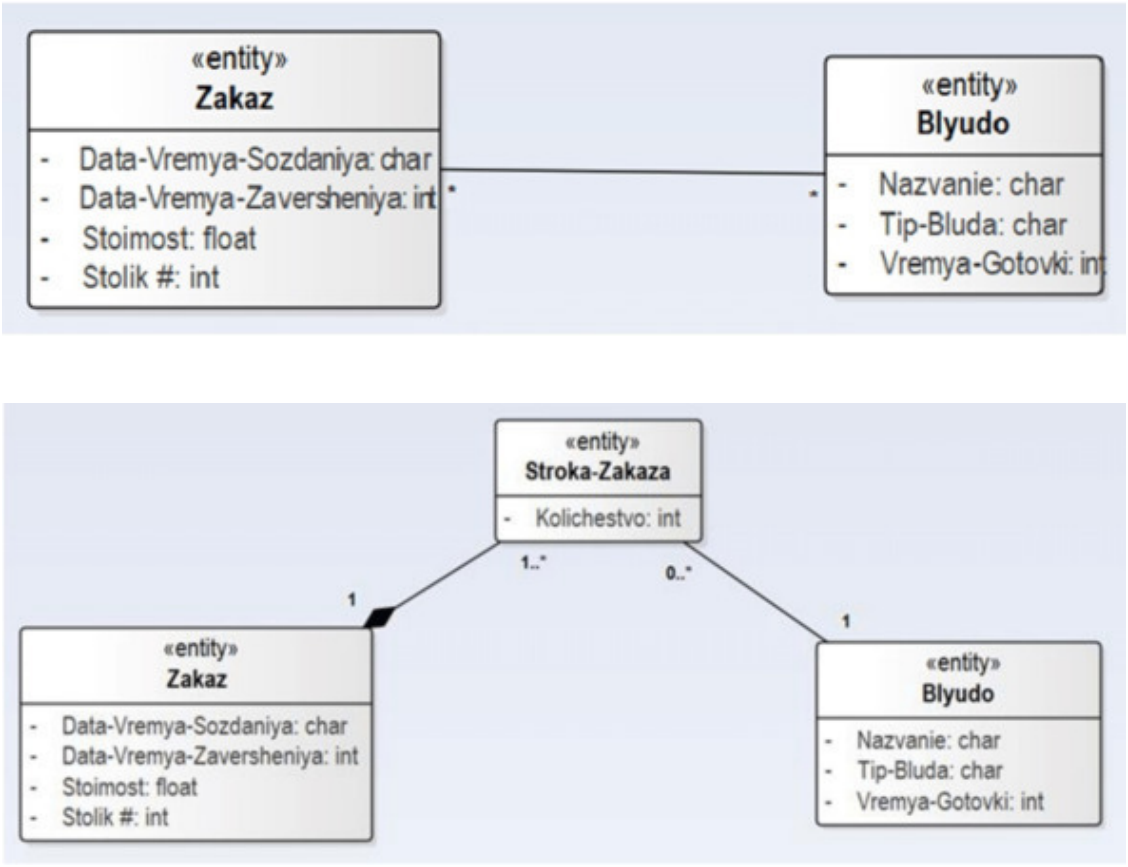

Рис. 1.11. Диаграмма Классов «Заказ» преобразовывается по паттерну «Объект – список», который применен к ассоциации «Заказ» – «Блюдо».

«Строка заказа» связана с «Заказом» композицией (закрашенный ромбик) – это значит, что при удалении «Заказа» все его «Строки заказа» также будут удалены.

Применим паттерн «Объект список» к ассоциациям «Много ко многим» для оставшихся событий «Закупка продуктов», «Списание бракованных продуктов», «Списание бракованных блюд» и «Инвентаризация», «Меню», а также к справочнику «Блюдо» – соответствующие диаграммы классов приведены ниже, на рис. 1.12—1.16.

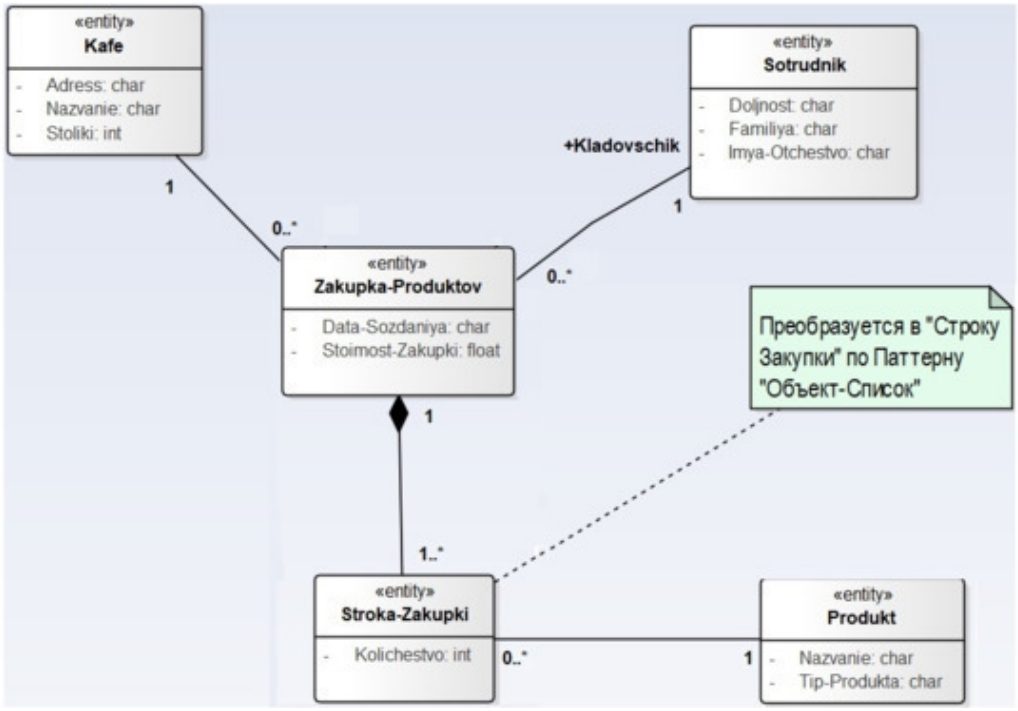

Рис. 1.12. Диаграмма классов «Закупка продуктов» преобразуется по паттерну «Объект – список» – сравните с рис. 1.4

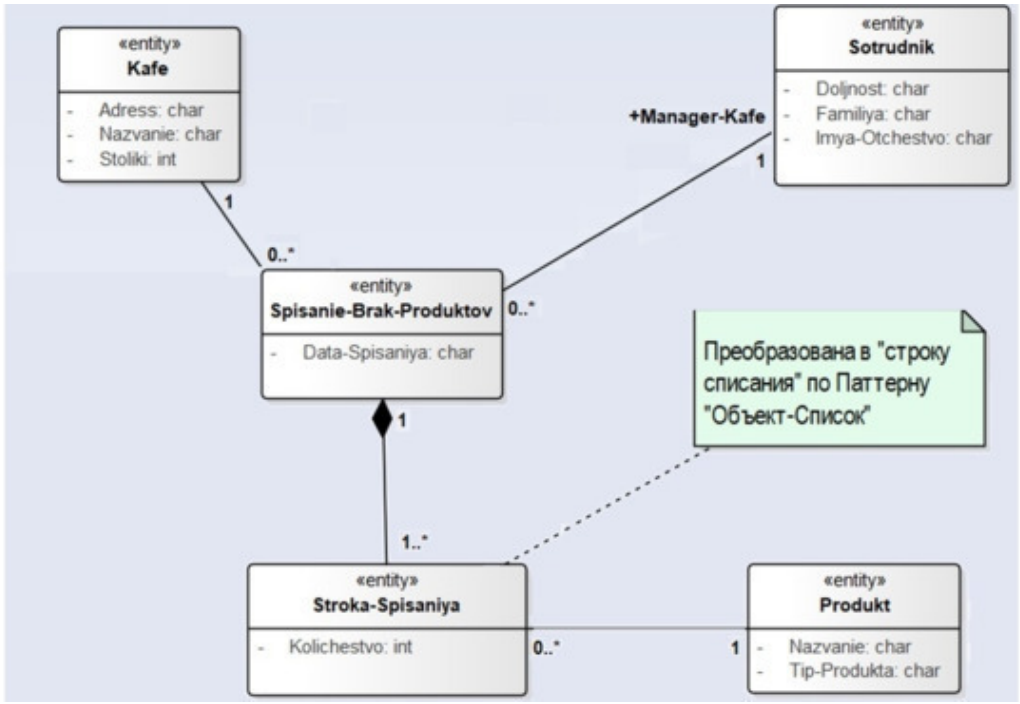

Рис. 1.13. диаграмма классов «списание бракованных продуктов» преобразуется по паттерну «объект – список» – сравните с рис. 1.5

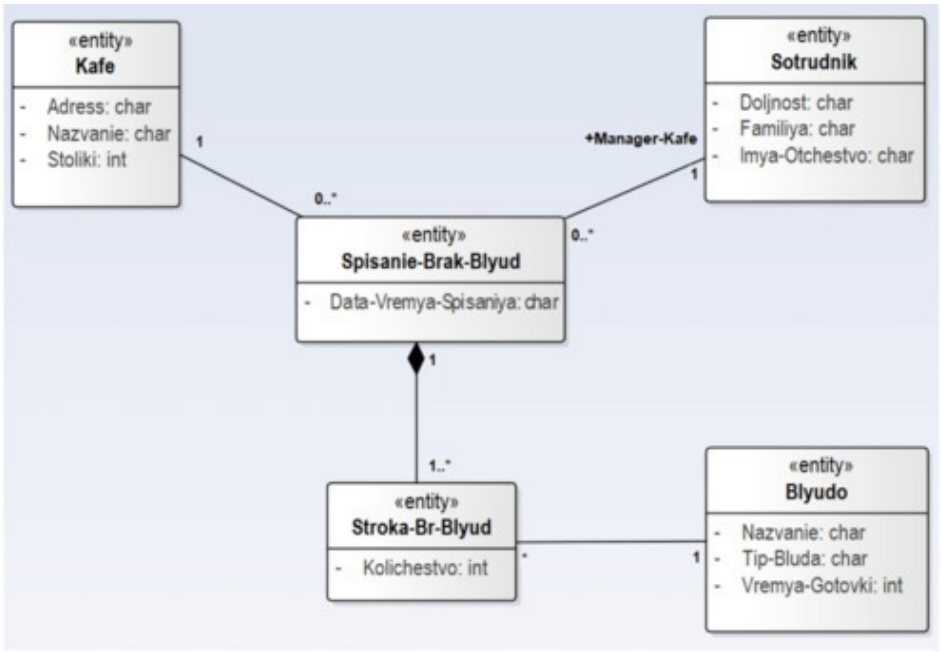

Рис. 1.14. Диаграмма классов «Списание бракованных блюд» преобразуется по паттерну «Объект – список» – сравните с рис. 1.6

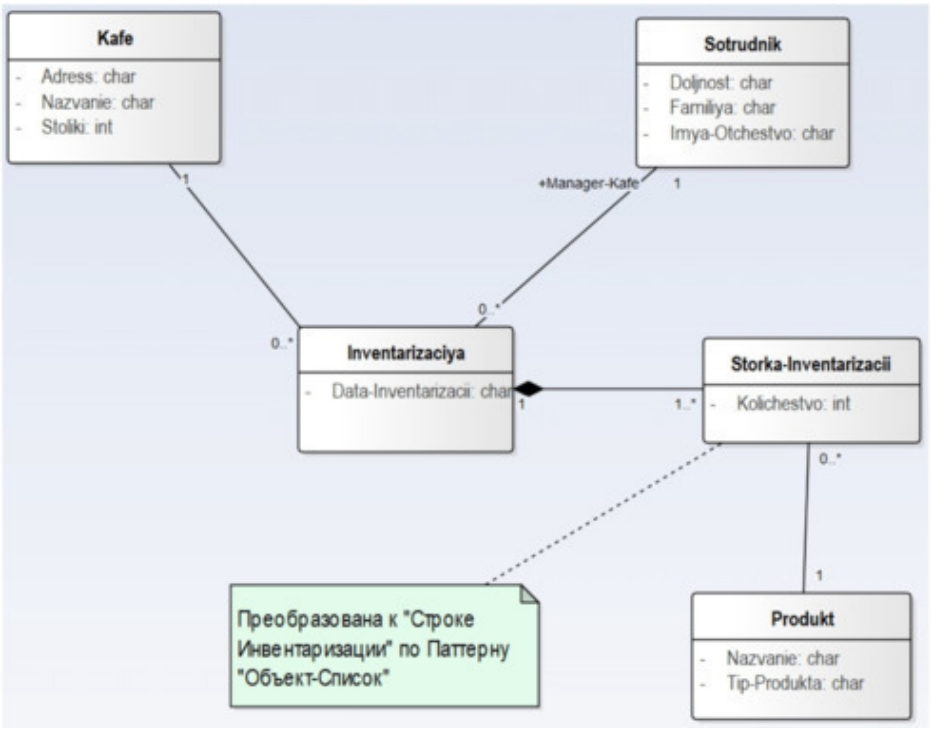

Рис. 1.15. Диаграмма классов «Инвентаризация» преобразуется по паттерну «Объект – список» – сравните с рис. 1.7

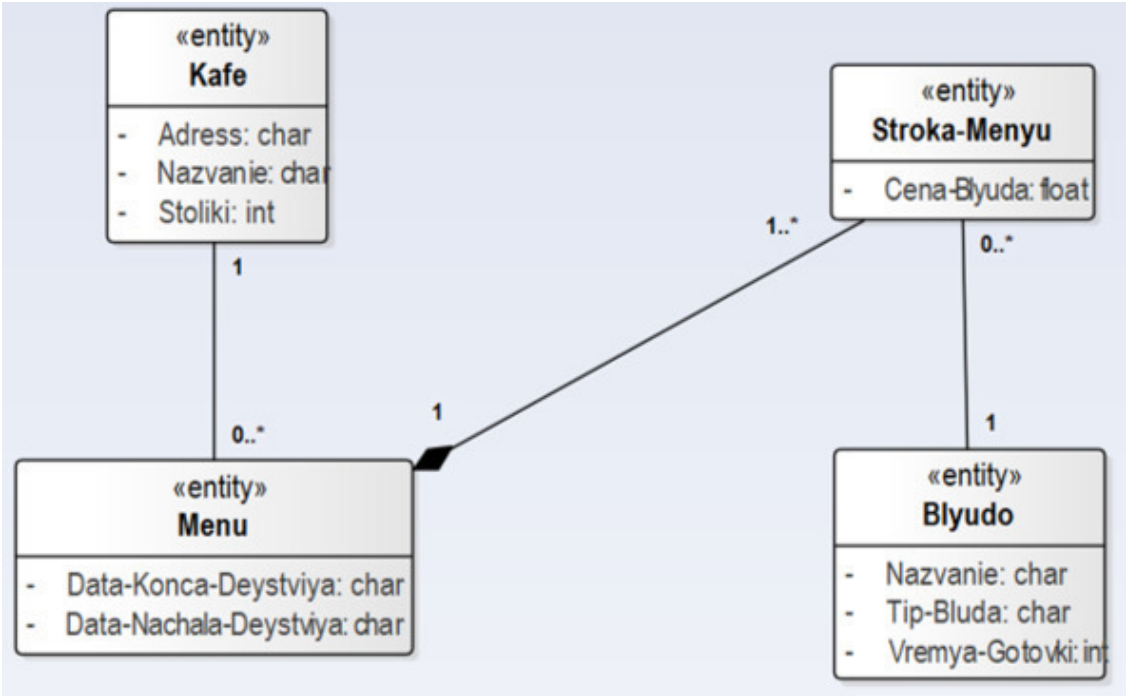

Рис. 1.16. Диаграмма классов «Меню» преобразуется по паттерну «Объект – список» – сравните с рис. 1.8

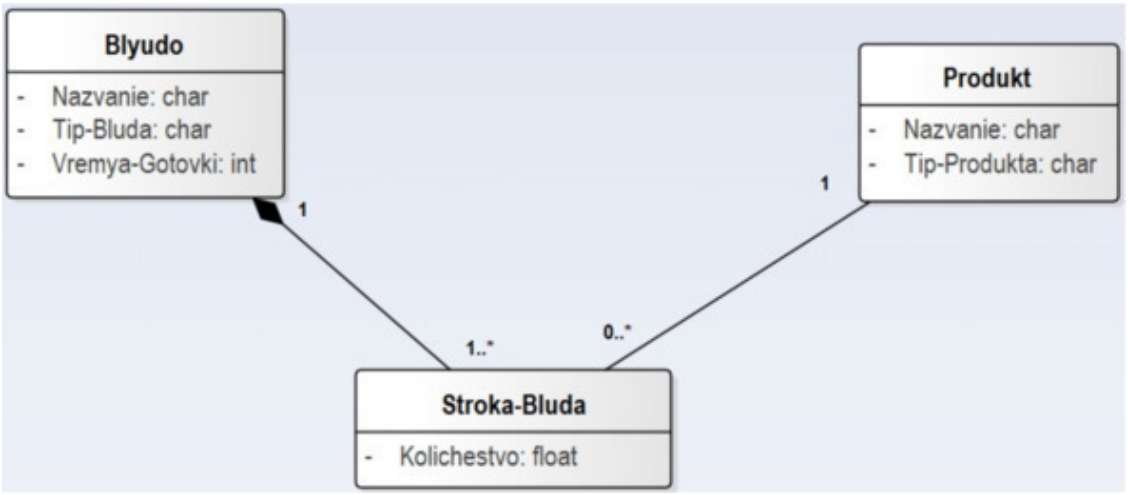

Рис. 1.16. Диаграмма классов «Блюдо» преобразуется по паттерну «Объект – список» – сравните с рис. 1.10

Паттерн «Объединение картотек»

Если на диаграмме классов встречаются ассоциации с такой множественностью, как:

• «один» к «одному» («1» – «1»),

• «один» к «ноль..одному» («1» – «0..1»),

• «ноль..один» к «ноль..одному» («0..1» – «0..1»), то соответствующие картотеки можно рассматривать как «*кандидаты на объединение*». В нашем примере такая связь есть на диаграмме «Заказ» между классом «Заказом» и классом «Оплатой заказа». Если мы объединяем эти картотеки, то:

• объединяются атрибуты картотек;

• новый класс (картотека) приобретает ***состояния.***

Для диаграммы классов «Заказ» получаем следующую преобразованную диаграмму классов уже без связи с «Оплатой заказа» (рис. 1.17).

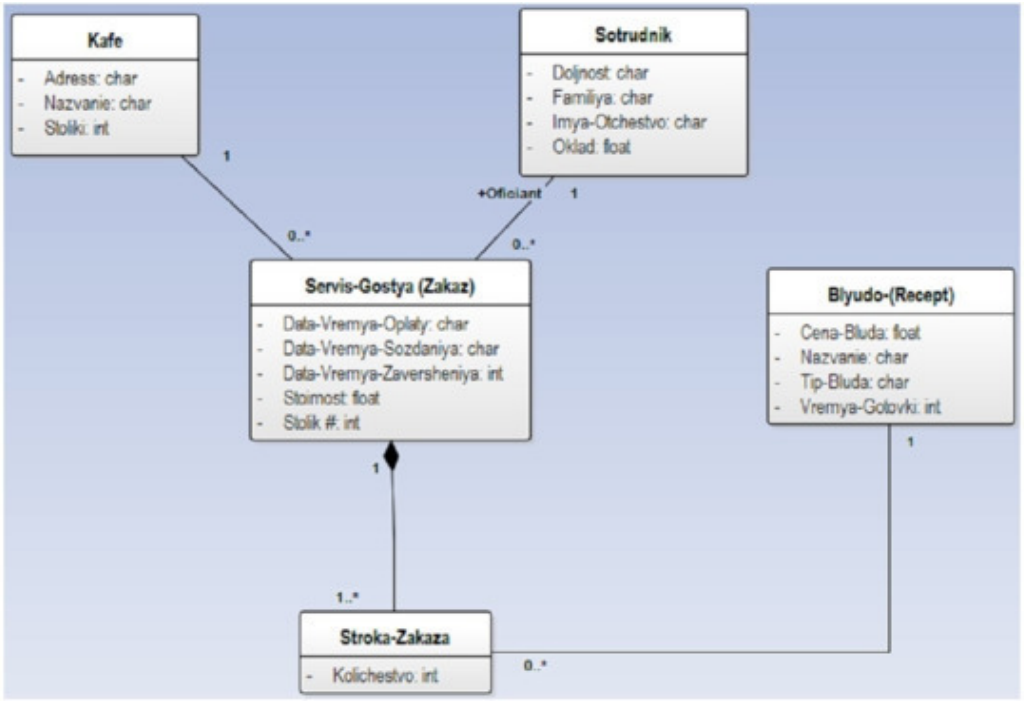

Рис. 1.17. Диаграмма классов «Заказ» после применения паттерна «Объединение картотек». У «заказа» появился новый атрибут «Дата-Время-Оплаты»

Состояния объединенной картотеки «Заказ» опишем UML-диаграммой «Машина состояний» (рис. 1.18).

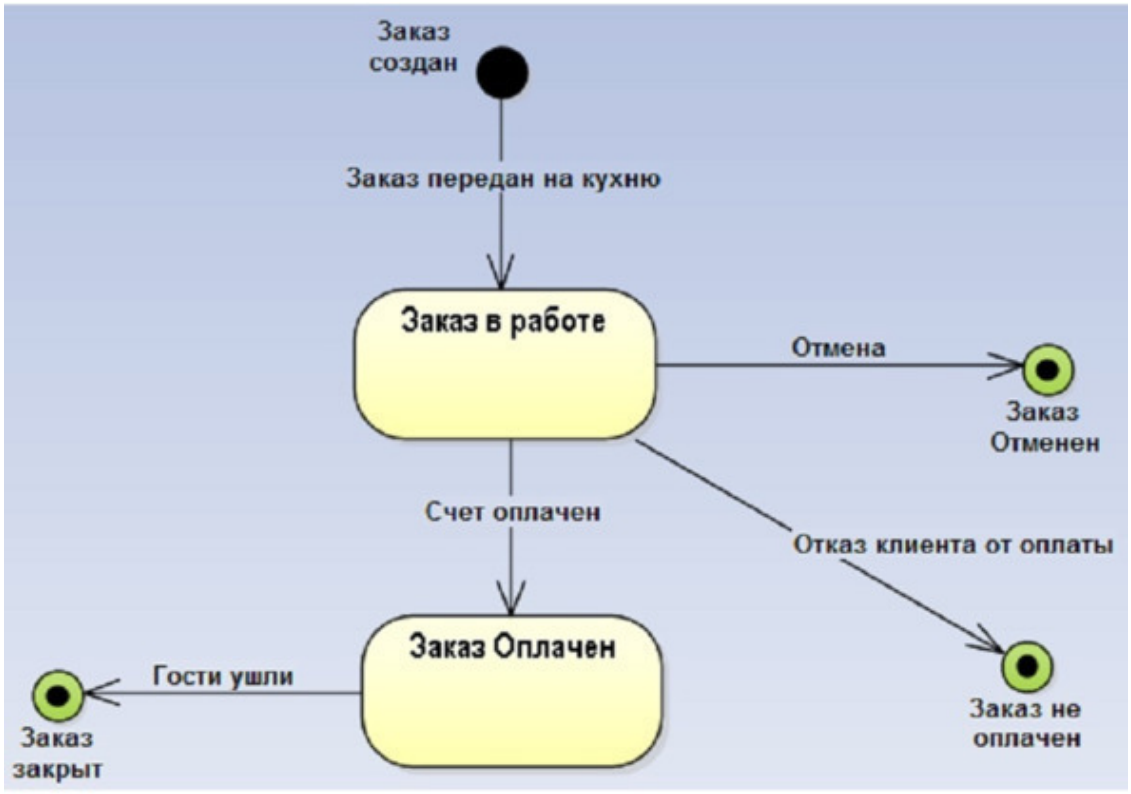

Рис. 1.18. Диаграмма «Машина состояний UML для класса „Заказ“ после объединения с картотекой „Оплата заказа“»

Итоги раздела 1

Был рассмотрен пошаговый процесс построения модели предметной области в виде набора диаграмм классов в визуальной нотации UML. Были рассмотрены паттерны *выявления бизнес-событий* «Брак», «Инвентаризация» и «Прайс-лист».

После построения диаграмм классов было показано, как и когда применяются паттерны преобразования модели – паттерны «Объект – список» и «Объединение картотек».

Предлагается применить описанный выше процесс к другим задачам, постановка которых приведена в приложении 1: «Театральные кассы», «Автоматизация поликлиники», «Таксопарк», «Мастерские автообслуживания», «Информационные материалы», «Документы муниципалитета», «Риелторская контора», «Расчет зарплаты», «Регистрация студентов на курсы».

Раздел 2

Требования к системе

В этом разделе будет показано, как формировать функциональные требования к ИС, опираясь на модель предметной области. Изложение будет построено на основе понятия сценария использования, которое является базовым в методологии разработки программного обеспечения IBM RUP (IBM Rational Unified Process)[6]. В свое время это была одна из ведущих методологий, и если процессы разработки в организации ставились на основе RUP, то это обеспечивало прозрачность и управляемость проекта, т. е. позволяло создавать ИС в соответствии с требованиями заказчика на момент сдачи системы.

RUP определяет роли, задачи, артефакты и 9 процессов, применение которых помогает сделать разработку программного обеспечения предсказуемой. IBM RUP – это оформленная в виде web-сайта электронная энциклопедия «правильной» (Framework) разработки, которая описывает основные процессы создания ИС:

1. Моделирование бизнес процессов.
2. Управление требованиями.
3. Анализ и проектирование.
4. Реализация.
5. Тестирование.
6. Развертывание.
7. Управление изменениями и конфигурациями.
8. Управление проектом.
9. Управление средой разработки.

Далее будем рассматривать методы и технику работы с требованиями к ИС на основе соответствующей «дисциплины» IBM RUP.

Было замечено, что в проектах разработки ИС часто возникают очень похожие друг на друга проблемы:

- Формально ИС удовлетворяет требованиям, однако заказчик не доволен: цели пользователей ИС или бизнес-цели не достигнуты.
- Требования запутаны, постоянно дополняются и изменяются.
- Модули не интегрируются при сборке системы.
- Затруднено сопровождение готового продукта.
- Дефекты, особенно в требованиях, обнаруживаются слишком поздно в проекте.
- Низкое качество рабочих материалов, создаваемых в проекте.
- Низкая производительность ИС при реальной нагрузке.
- Действия в рамках команды плохо скоординированы.

Анализ успешного опыта команд, преодолевших подобные проблемы, позволяет выделить практические паттерны – лучшие практики (Best Practices), которые могут быть использованы в любых проектах, связанных с разработкой ПО.

Rational Unified Process основан на шести лучших практиках:

- итеративная разработка;

[6] Крачтен Ф. Введение в Rational Unified Process.: пер. с англ. М.: Вильямс, 2002. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения.: пер. с англ. СПб: Питер, 2002.

• управление требованиями;
• использование компонентной архитектуры;
• визуальное моделирование;
• постоянный контроль качества;
• управление изменениями и конфигурациями.

Преимущества от использования лучших практик в проекте разработки программного обеспечения состоят в следующем:

• разработка системы ориентирована на удовлетворение бизнес-потребностей заказчика в автоматизации;

• лучшие практики при их совместном применении усиливают положительные эффекты друг от друга;

• вся команда хорошо понимает, что делать, как делать и когда делать.

Две из этих практик – «Управление требованиями» и «Визуальное моделирование» – мы рассматриваем в книге.

Визуальное моделирование позволяет:

• понять **структуру** создаваемой системы, которая разбивается на части, компоненты, элементы;

• понять **поведение** создаваемой системы – как элементы взаимодействуют друг с другом при выполнении заданных операций;

• наглядно показать **взаимосвязи элементов** системы друг с другом;

• обеспечить **целостность и согласованность** проекта с реализацией системы;

• **документировать** принятые проектные решения, включая вновь возникающие запросы на изменения;

• **скрывать или показывать** детали элементов в зависимости от назначения модели и потребностей ее будущих «читателей»;

• обеспечить **однозначность коммуникаций** внутри команды, которые основаны на использовании моделей системы как «ее чертежей».

Одним из средств для построения визуальных моделей является Unified Modeling Language (UML) – «графический язык», обеспечивающий описание статических и динамических аспектов системы. Методология IBM RUP основана на повсеместном использовании визуальной нотации UML.

Модель описания требований «FURPS+»

Выявление требований согласно модели **FURPS+** позволяет создать качественный набор документов в проекте. Аббревиатура FURPS образована из характеристик:

• **Functionality** – функциональность;
• **Usability** – удобство использования;
• **Reliability** – надежность;
• **Performance** – производительность;
• **Supportability** – удобство сопровождения.

При этом часть формулировок требований относится к ограничениям на проектирование, реализацию и интерфейсы (значок «**+**»):

• **Design** – ограничения проектирования;

• **Implementation** – ограничения на программную реализацию, например, разработка на заданном языке программирования;

• **Interface** – ограничения на интерфейсы;

Основной формой представления функциональных требований к ИС являются «сценарии использования» (Use-cases).[7] Модель сценариев использования (Use-case Model) представляет собой набор текстовых описаний для каждого выявленного сценария. Шаги сценария со стороны системы представляют собой функциональные требования к ИС. Сценарий использования (Use-case) – это ключевое понятие RUP, которое является единицей организации работ в проекте (т. е. это единица *планирования*, *проектирования*, *отладки и тестирования* ИС). Модель сценариев использования является основой для эффективного ***планирования работ*** и управлению проектом, без его использования эффективность работы в проекте существенно снижается.

Требования и их документирование

Основными видами артефактов, содержащих описания требований к системе, согласно IBM RUP являются:

• запросы заинтересованных лиц (ЗЛ – Stakeholder Requests);
• глоссарий (Glossary).
• видение (Vision);
• модель сценариев использования (Use Case Model, состоит из Use Case Specification);
• дополнительные спецификации (Supplementary Specification).

В приложении 1 приведены шаблоны этих документов с кратким пояснением разделов.

[7] Известны названия-синонимы: «варианты использования» или «прецеденты».

Последовательность обработки требований

1. Выявить заинтересованных лиц (stakeholders, список ведется в Vision).

2. Собрать их пожелания (набор документов – Stakeholder Requests).

3. Выделить (путем символьной разметки) в пожеланиях **ключевые потребности** (needs) и классифицировать их:

• симптомы бизнес-проблем («issues»);

• действующие лица (пользователи, «needs-actors»);

• варианты использования («need-use cases»);

• бизнес-правила и бизнес-процессы («needs-business rules», «needs-business processes»);

• ограничения (needs-constraints).

4. Основные термины проекта описать в Глоссарии (Glossary), используя модель предметной области.

5. Проанализировать симптомы и сформулировать основную бизнес-проблему, на решение которой будет направлена разрабатываемая ИС (секция Problem Statement формулируется в Vision).

6. Отобрать потребности (Needs), соответствующие предложенному направлению решению проблемы (Problem Statement используется как фильтр).

7. Для отобранных потребностей (на основе Глоссария) сформировать перечень бизнес-требований к разрабатываемой системе (Features формулируется в Vision). Построить матрицу трассировки Needs и Features.

8. На основе модели предметной области выявить сценарии использования. Построить UseCases диаграмму на UML.

9. Описать эскизы (Outline) сценариев использования (UseCases), трудоемкость которых можно оценить по объему альтернативных потоков.

Конец ознакомительного фрагмента.