А. А. ЦВЕТКОВ

Теория и практика бизнес-анализа в ИТ

Том II

Учебное пособие

ИНСТИТУТ ПРОГРАММНЫХ СИСТЕМ РАН

А. А. Цветков

Теория и практика бизнес-анализа в ИТ

Том II

Учебное пособие

DirectMEDIA

Москва
Берлин
2020

УДК 339.1:658(075)
ББК 65.29я7

Решением Ученого Совета Института программных систем им. А. К. Айламазяна РАН учебное пособие рекомендовано к печати

Рецензенты:
Александрова Ирина Алексеевна, кандидат технических наук;
Непейвода Николай Николаевич, доктор физико-математических наук, профессор

Цветков, А. А.
Е 67 Теория и практика бизнес-анализа : учебное пособие. В 2 т. Т. II / А. А. Цветков. – Москва ; Берлин : Директ-Медиа, 2020. – 99 с. : ил.

ISBN 978-5-4499-0006-7

Настоящая книга «Теория и практика бизнес-анализа в ИТ. Том II» является продолжением описания особенностей работы бизнес-аналитика, изложенных в I-м томе.

Приводится подробное описание инструментов бизнес-аналитика, которые не вошли в I-й том: нотация унифицированного языка моделирования (UML) в объеме, который необходим бизнес-аналитику (модели «Варианты использования» и «Диаграмма последовательностей»), и описание особенностей моделирования пользовательского интерфейса. Нотации приводятся в соответствии с оригинальными документами Object Management Group (OMG), т. е. документами международного консорциума, на основе которых формируются международные стандарты ИСО/МЭК.

Раздел книги, посвященный UML, снабжен заданиями для повторения и усвоения изученного материала.

В приложении приводится справочник по нотации UML.

В основу книги положен практический опыт разработки аналитических моделей автором и его коллегами, а также курс лекций, читавшийся студентам в рамках курса «Информационные системы».

Настоящая книга рекомендована в качестве учебного и справочного пособия для студентов магистратур и аспирантов, которые обучаются по специальностям, связанным с информатикой и смежными дисциплинами, в соответствии с решением Ученого Совета Института программных систем им. А. К. Айламазяна РАН от «12» 2018 г.

УДК 339.1:658(075)
ББК 65.29я7

ISBN 978-5-4499-0006-7

Оглавление

ПРЕДИСЛОВИЕ АВТОРА КО ВТОРОМУ ТОМУ

В предыдущем томе были изложены основные понятия, описывающие бизнес-анализ (далее – БА), которые включали понимание того, для чего БА проводится, т. е. введение в разработку информационных систем (далее – ИС), а также описаны информационные бизнес-процессы (далее – ИБП), которые выполняются в процессе разработки ИС. В качестве плана действий[1], который используется при разработке, использовалась диаграмма Захмана, доказавшая на протяжении десятков лет свою корректность и эффективность. Основной упор делался на ИБП бизнес-аналитика, включая его взаимодействие с остальными участниками проектной группы и заказчиком ИС.

Можно сказать, что бизнес-аналитик является двуликим Янусом: одно лицо обращено к заказчику, а другое – к проектной группе, но мозг один, в задачи которого входит понять то, ЧТО хочет заказчик, решить, КАК должна выглядеть ИС, чтобы воплотить мечты заказчика (насколько это возможно) в действительность, отразить свое решение в виде, который позволит проектной группе реализовать эти «ЧТО» и «КАК». По сути, бизнес-аналитик – это внутренний заказчик, который постоянно взаимодействует со своими коллегами в процессе разработки ИС, но для заказчика – это представитель разработчика, который вместе с руководителем проекта получает за всю группу либо восторги, либо негатив.

Для того, чтобы документы, которые разрабатывает бизнес-аналитик, были максимально прозрачны для понимания, они должны содержать два компонента: текстовый и графический. При этом графический компонент – это не произвольная картинка, а модель, которая формируется в соответствии со стандартными нотациями, часть из которых была рассмотрена в предыдущем томе: нотация модели бизнес-процессов (далее – BPMN) и модель «Сущность-связь» (далее – ERD).

---

[1] Автор категорически против прямых калек с англоязычной литературы, которые засоряют и искажают русский язык. В частности, такое словосочетание, как «дорожная карта». Зачем? Чем не устраивает наш родной «план действий»? Давайте беречь родной язык.

BPMN используется главным образом для того, чтобы отразить знания, полученные от заказчика, который на этапе предпроектного обследования выступает в качестве носителя знаний в предметной области (далее – ПрО) для создаваемой ИС. Бизнес-аналитик в данном случае выполняет роль инженера по знаниям, т. е. специалиста, который может адекватно формализовать получаемые знания. Формализация знаний в графическом виде выполняется именно посредством нотации BPMN: первая итерация – набор моделей «Как есть», вторая итерация – набор моделей «Как будет».

Кроме того, формируется концептуальная модель ERD, отражающая сущности ПрО.

Эти два вида моделей нужны для того, чтобы, во-первых, бизнес-аналитик вник в ПрО до уровня, когда он начинает чувствовать себя в ней, если не экспертом, то, по крайней мере, знатоком, а, во-вторых, чтобы заказчик мог оценить насколько бизнес-аналитик верно интерпретировал знания и отразил пожелания по автоматизации ИБП.

Теперь перед бизнес-аналитиком стоит задача сформировать задание для группы разработки так, чтобы это задание однозначно описывало видение ИС бизнес-аналитиком (или внутренним заказчиком). От того, насколько тщательно, однозначно и понятно сформировано задание, зависит реализация ИС или, в конечном счете, получит ли ожидаемый результат заказчик. Если «тщательность» – это скорее понятие, характеризующее поведенческие особенности бизнес-аналитика, то «однозначность» и «понятность» относятся к категории соблюдения стандартов и другой нормативно-справочной информации (далее – НСИ).

Во втором томе рассматривается нотация универсального языка моделирования (далее – UML) в том объеме, которым должен владеть бизнес-аналитик[2], описывается формализация

[2] Никто не мешает знать UML в полном объеме – приходится ограничиваться в данной книге, т. к. полное описание UML потребовало бы отдельной книги, а может и не одной…

пользовательских сценариев и формирование концептов форм, с которыми должен взаимодействовать конечный пользователь.

Выражаю благодарности своим коллегам и друзьям, которые поддерживали меня при написании этого двухтомника.

Но, в первую очередь, огромное спасибо жене Елене, которая терпеливо и философски относится к моим творческим мучениям.

ГЛАВА 1.
НОТАЦИЯ УНИФИЦИРОВАННОГО ЯЗЫКА МОДЕЛИРОВАНИЯ (UML)

- **Введение в UML**
- **Модель «Вариант использования» (Use Case Diagram)**
- **Модель «Последовательность» (Sequence Diagram)**
- **Выводы**

A. Введение в UML

Когда бизнес-аналитик общается с заказчиком, то он, фактически, становится специалистом в ПрО, для которой будет разрабатываться ИС, т. е. начинает оперировать терминами и понятиями заказчика. А как иначе – трудно предположить, что специалист, например, в области торговли обувью или в области управления полетами, будет еще и владеть профессиональными знаниями в области разработки и создания ИС (это было бы настоящим чудом). Да ему это и не нужно – вряд ли кто задумывается над законами физики, когда включает утюг или телевизор: работает и слава Богу, большое спасибо тем, кто это придумал. С другой стороны, специалисты в области информационных технологий (далее – ИТ) могут (но не факт) хорошо знать свою специальность, но ничего не понимать, например, в металлургии. Отсюда возникает понимание важности роли бизнес-аналитика, который должен интерпретировать видение заказчика на формализованный язык, понятный ИТ-специалистам.

Существует ли такой формализованный язык? Ответ – ДА. Это UML и знать его ОБЯЗАН каждый член группы разработки. К сожалению, автору слишком часто приходится встречаться с ситуацией, когда в группе программистов хорошо знают какой-либо язык программирования, но не умеют читать модели UML. Те, кто попроще, спросят, ткнув пальцем в модель: «Это чё за хрень?», те, кто похитрее, ничего не скажут, но разра-

ботают нечто, что зачастую полностью расходится с проектом. А дальше… Из-за необходимости переделок срываются сроки разработок и начинаются авральные работы, нервотрепка и т. п. «прелести», как следствие невежества. Еще одна неприятная ситуация – члены группы разработки «как бы» знают UML, но, как-то по-своему, т. е., например, в соответствии с нотацией стрелка показывает, что этот модуль входит в состав некоторого агрегата (элемент «включение» или, если по-английски «include» (см. описание элемента далее)), а интерпретация «грамотея» будет такой: агрегат передает данные или управление модулю. Но вся проблема в том, что модуль – это часть агрегата… Если обратиться к машиностроению, то это будет равносильно декларации, что шестеренка, входящая в механизм, живет собственной насыщенной жизнью. Без комментариев.

Разработчики ИС рассматривают ее с разных сторон в соответствии с их ролью в группе. Это могут быть, например:

- аналитики;
- дизайнеры;
- программисты;
- тестировщики;
- специалисты по качеству;
- технические писатели и др.

Для выполнения своих функций, каждому из специалистов нужно свое представление моделей ИС. Существует известная шутка: у руководителя проекта спросили почему документация содержит двадцать комплектов моделей системы, неужели система настолько сложная; на что он ответил – все потому, что в группе работает двадцать аналитиков.

Именно необходимость представления ИС в виде множества моделей, каждая из которых отражается специфический взгляд на систему, привела к тому, что в 1997 году появилась первая спецификация нотации UML, которая получила дальнейшее развитие и в настоящее время существует версия 2.5.1 (см. [3]).

Парадигма «Представление ИС с разных сторон» в нотации UML реализована путем разбиения видов моделей на две большие группы: структурные модели (англ. «Structure diagrams»),

которые отражают статическую структуру ИС, и поведенческие модели (англ. «Behavior diagrams»), которые отражают динамику поведения объектов в ИС:

- Структурные модели:
  - модель классов (англ. «Class Diagram»);
  - модель компонентов (англ. «Component Diagram»);
  - модель объектов (англ. «Object Diagram»);
  - модель развертывания (англ. «Deployment Diagram»);
  - модель пакетов (англ. «Package Diagram»);
  - модель композитной структуры (англ. «Composite Structure Diagram»);
  - модель профиля (англ. «Profile Diagram»);
- Поведенческие модели:
  - модель вариантов использования (англ. «Use Case Diagram»);
  - модель деятельности (англ. «Activity Diagram»);
  - модель конечного автомата (англ. «State Machine Diagram»);
  - модель последовательности (англ. «Sequence Diagram»);
  - модель коммуникаций (англ. «Communication Diagram»);
  - обзорная модель взаимодействий (англ. «Interaction Overview Diagram»);
  - временная модель (англ. «Timing Diagram»).

Не все типы моделей, из перечисленных выше, требуются бизнес-аналитику для решения задач, стоящих перед ним. Напомним какие вопросы стоят перед бизнес-аналитиком в процессе взаимодействия с функциональным заказчиком и какую информацию необходимо передать системному аналитику[3]

[3] «Передать системному аналитику» или, как сейчас стало модно говорить, системному архитектору – это справедливо для очень больших проектов, в которых в группе разработки существует такая роль. Желательно и для средних по масштабу проектов, но не всегда возможно. Во многих случаях роли бизнес-аналитика и системного аналитики совмещает один человек или просто – аналитик. Настоятельно рекомендую: даже в том случае, если вы совмещаете обе роли, готовить данные так, словно информация будет передана другому человеку. Это дисциплинирует. Например, такой педантичностью отличался швейцарский физик-теоретик Ф. Э. Паули,

(а через него и всей группе разработки) для того, чтобы разрабатываемая ИС соответствовала ожиданиям заказчика и была реализована так, как это предусмотрел бизнес-аналитик, т. е. передаваемая информация должна однозначно отражать его видение, согласованное с функциональным заказчиком. В соответствии с диаграммой Захмана, ответы на вопросы: «ЧТО», «КАК», «ГДЕ», «КТО», «КОГДА», «ПОЧЕМУ», которые формирует бизнес-аналитик, передаются системному аналитику. Что это за ответы и в каком виде они должны быть сформированы?

Ответ на вопрос «ЧТО» мы достаточно подробно рассмотрели в первом томе настоящего учебного пособия – это модели ERD. Т. е., согласно диаграмме Захмана, бизнес-аналитиком должен быть подготовлен список бизнес-сущностей и связей (или документ, который можно условно назвать «Учет представлений») в виде концептуальной модели в соответствии с нотацией ERD и соответствующим текстовым описанием, которое дополняет и однозначно интерпретирует модель данных. Согласно рекомендациям (или требованиям, если компания-разработчик твердо стоит на позиции соблюдения ГОСТов) ГОСТ серии 34*, а именно документу РД 50–34.698–90 «Методические указания. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов», список бизнес-сущностей и связей или ссылка на документ, который содержит этот список, должен быть включен в документ «Пояснительная записка» в раздел «Основные технические решения».

Ответы на вопросы «КАК», «ГДЕ», «КТО», «КОГДА», «ПОЧЕМУ» тоже рассматривались в первом томе настоящего учебного пособия в виде моделей в нотации BPMN. Но эти модели больше ориентированы на то, чтобы, с одной стороны, заказчик мог убедиться в том, что вы, как бизнес-аналитик, правильно поняли процессы, существующие у заказчика, и корректно отразили будущее бизнес-архитектуры

который в ответственных случаях писал подробное задание самому себе и отправлял это задание на свой адрес по почте. Конечно чудачество, но что-то в этом есть…

организации, включив туда автоматизацию ИБП. Из моделей BPMN разработчикам ИС может быть не всегда однозначно ясно, например, какие подсистемы должны входить в проектируемую ИС (границы проекта), какие пользователи (группы пользователей) будут взаимодействовать с ИС и какие права у них будут в ИС, в какие временные интервалы должны укладываться подсистемы при выполнении функций, возложенных на них, какие внешние подсистемы предполагается использовать и множество других вопросов, на которые не могут ответить модели в нотации BPMN или использование этой нотации приведет к сложностям в интерпретации модели. Т. е. задачей бизнес-аналитика является интерпретация моделей в нотации BPMN в нотацию UML (поведенческие модели). Степень детализации моделей должна позволить на этапе системного анализа сформировать однозначные задания программистам в виде формализма, не предполагающего изучения ПрО заказчика со стороны остальных членов группы.

Далее мы рассмотрим модели UML, которые будут отвечать на поставленные вопросы, но уже с точки зрения проектирования ИС, являясь продолжением видения бизнес-аналитика на языке, который понятен системному аналитику и позволяет ему формировать задачи для программистов.

B. Модель «Варианты использования» (Use Case Diagram)

Модель «Варианты использования» (далее – UCD) позволит нам ответить на вопросы «КАК», «ГДЕ», «КТО», но не ответит на остальные вопросы напрямую – только через подробнейшие текстовые комментарии в самой модели или в сопровождающем тексте. Собственно, главная задача этого вида моделей показать варианты использования проектируемой ИС или, если в буквальном смысле: «КТО» и «КАК» выполнит некоторое действие и «ГДЕ» будут располагаться объекты, участвующие в выполнении действия. Т. е. можно сказать, что UCD своеобразный «язык» (впрочем, как любая нотация моделирования), который имеет свои «подлежащие», «сказуемые» и др. элементы естественных языков. Элементы нотации немного-

численны, что позволяет достаточно лаконично отобразить суть моделируемых вариантов использования.

Элементом нотации UCD, отвечающим на вопрос «КТО» (или одно из «подлежащих»), является «Исполнитель» (англ. *Actor*), который обычно принято изображать в виде схематического изображения человечка (см. Рисунок 1), но нотация UML допускает и другие варианты отображения элемента «Исполнитель», которые в каждой конкретной ситуации будут точнее отображать смысл, передаваемый графическим элементом.

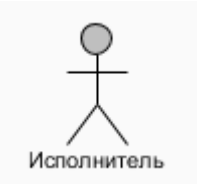


Рисунок 1 – Элемент нотации «Исполнитель» в виде человечка

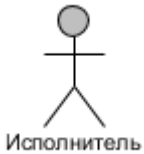
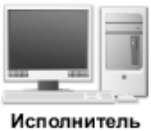
Согласно нотации UML, для элемента «Исполнитель» подразумевается, что:

- это одушевленный или неодушевленный объект, который взаимодействует с элементом «Вариант использования» (системной функцией) (см. ниже);
- наименование элемента должно быть именем существительным;
- объект выполняет некоторую деятельность;
- элемент «Исполнитель» аналогичен понятию «пользователь», но может иметь несколько ролей (например, сущность «доцент» в реальной жизни может быть и преподавателем, и исследователем);
- элемент «Исполнитель» «запускает» элемент «Вариант использования» (системную функцию) (см. ниже);
- элемент «Исполнитель» воздействует на систему через входы и ожидает реакции системы через выходы.

Как отмечено выше, допускается использование иных графических элементов для отображения элемента «Исполнитель» (см. Таблица 1). Допускаются любые иные графические

элементы, которые могут подчеркнуть особенности элемента «Исполнитель» в контексте модели, и, по использованию которых достигнуто соглашение, как в группе разработки, так и согласовано с заказчиком ИС.

Таблица 1 – Примеры допустимого графического представления элемента «Исполнитель» (графика Visual Paradigm)

Графическое представление элемента «Исполнитель»	Примечание
	**Примитивная форма**. Может использоваться во всех случаях, например, если графический редактор не позволяет использовать другие представления
	**Человечек**. Наиболее употребляемое представление элемента «Исполнитель». Подходит практически для всех ролей, соотнесенных с изображением. В настоящей книге будет использоваться для отображения всех одушевленных объектов
	**Рабочий стол**. Представление элемента «Исполнитель» в виде, который подчеркивает, что исполнитель взаимодействует с системой через приложение на компьютере
	**Приложение**. Представление, которое подчеркивает, что некоторое действие выполняется приложением, не входящим в систему
	**Сервер приложений**. Представление, которое подчеркивает, что с системой взаимодействует набор приложений, расположенных на сервере приложений

Графическое представление элемента «Исполнитель»	Примечание
 Исполнитель	**Облако**. Представление, которое подчеркивает, что «Исполнитель» расположен в облаке, т. е. не имеет значения природа взаимодействия с системой: это может быть, как одушевленный, так и неодушевленный объект

Модель UCD может отражать процессы, которые происходят в организации, как до автоматизации ИБП, так и после внедрения ИС. При этом среди исполнителей ИБП и будущих пользователей ИС существует разделение ролей, в соответствии с которыми они выполняют свои функции. Например, в некоторой гипотетической медицинской организации сотрудники образуют группы: «Сотрудники администрации», «Сотрудники лечебных отделений», «Сотрудники вспомогательных отделений». При этом все входят в обобщающую группу «Сотрудники медицинской организации»[4].

В нотации UML существует элемент, который позволяет отразить подобные структуры – это элемент «Обобщение», который изображается в виде стрелки, показанной ниже (см. Рисунок 2).

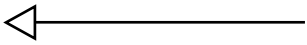


Рисунок 2 – Элемент нотации UML «Обобщение»

Элемент «Обобщение» отображает таксономическую связь между обобщающим классифицирующим элементом и классифицирующим элементом, отражающим специфику объекта, который располагается ниже в иерархии объектов[5].

---

[4] В действительности в состав большой медицинской организации входит гораздо больше подразделений, но для примера ограничимся тем, что приведено в тексте.

[5] Формирование моделей, описывающих некоторую ПрО, предполагает на первых этапах формирование семантической модели этой ПрО и составление глоссария терминов (см., например, [1]). Собственно, из семантической модели появляется понимание иерархии. Но обсуждение формирования семантических моделей выходит

Каждый классифицирующий объект может выступать в роли обобщающего для объектов, располагающихся ниже в классификации. При этом, каждый из классификаторов наследует свойства классификатора, расположенного выше. *Стрелка всегда направлена от менее общего объекта к более общему объекту.*

Модель для структуры сотрудников организации, приведенной выше, представлена ниже (см. Рисунок 3).

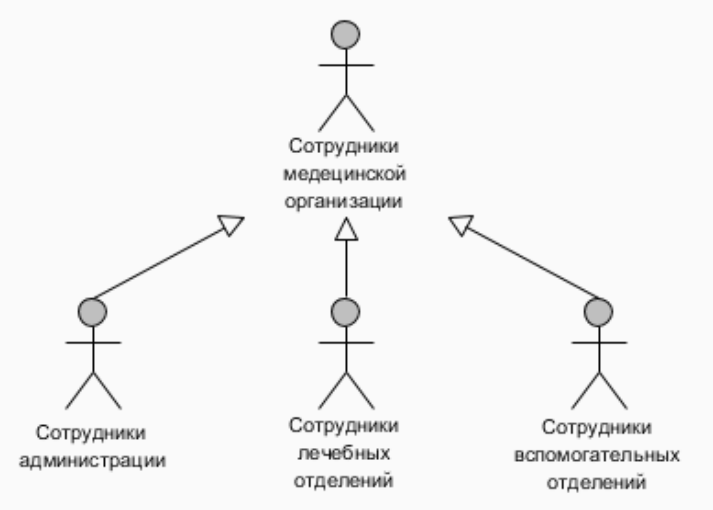


Рисунок 3 – Пример модели структуры сотрудников организации

Если, представленную модель, интерпретировать на естественном языке, предполагая, что элемент типа «Исполнитель» – это подлежащее, а элемент типа «Обобщение» – это сказуемое, то модель «зазвучит», например:

1. Множество «Сотрудники медицинской организации» включает в себя подмножества «Сотрудники администрации», «Сотрудники лечебных отделений», «Сотрудники вспомогательных отделений»; или

за рамки настоящего издания, а, поэтому автор надеется, что читателю будет интуитивно понятно иерархическое представление.

2. Подмножество «Сотрудники лечебных отделений» входит в множество «Сотрудники медицинской организации»; или …

Рассмотрим теперь, как отображается действие, которое выполняют «Исполнители», и, каждый из которых, согласно нотации UML, должен быть связан с каким-то процессом или функцией, выполняемой системой. Границы деятельности или границы системы в моделях UCD изображаются в виде прямоугольного элемента «Система», показанного ниже (см. Рисунок 4).

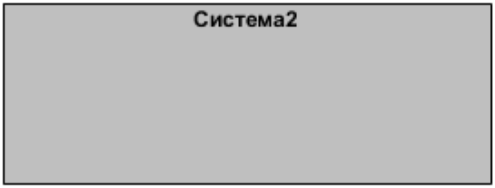


Рисунок 4 – Элемент нотации UML «Система»

Элемент «Система» содержит внутри элементы, которые отображают действия, выполняемые элементами «Исполнитель», и отвечает на вопрос «ГДЕ» находятся границы процессов до автоматизации или после автоматизации. Если процессы выполняются в разных местах или проектируемая ИС будет распределенной по разным площадкам, то на модели может существовать несколько элементов «Система».

Сами элементы, которые показывают, «ЧТО» происходит в границах деятельности или в системе, в нотации UML представляют в виде элемента «Вариант использования», который имеет вид, показанный ниже (см. Рисунок 5).

Рисунок 5 – Элемент нотации UML «Вариант использования»

Элемент «Вариант использования» должен отвечать следующим требованиям:

- Элемент представляет некоторую системную функцию (процесс может быть автоматическим или ручным);
- Наименование элемента «Вариант использования» должно быть глаголом в сочетании с существительным (или словосочетанием), т. е. обозначает некоторое действие;
- Каждый элемент «Исполнитель» должен быть связан с элементом «Вариант использования», но не каждый элемент «Вариант использования» соединяется с элементом «Исполнитель».

Элемент «Исполнитель» связывают с элементом «Вариант использования» посредством элемента «Ассоциация» (см. Рисунок 6) с тем, чтобы указать на факт использования «Исполнителем» данного сценария.

Рисунок 6 – Элемент нотации UML «Ассоциация»

Расширим наш пример с медицинской организацией следующим образом: добавим процессы, которые выполняют сотрудники. Повторю, что для примера делаются большие упрощения. Итак, процессы, выполняемые в гипотетической медицинской организации:

- Все сотрудники делают отметки в табеле о своем приходе/уходе на работу/с работы;
- Сотрудники администрации выполняют процессы администрирования;
- Сотрудники лечебных отделений выполняют процессы, связанные с лечебной деятельностью;
- Сотрудники вспомогательных отделений выполняют процессы по обеспечению деятельности других подразделений медицинской организации.

Вариант модели UCD представлен ниже (см. Рисунок 7).

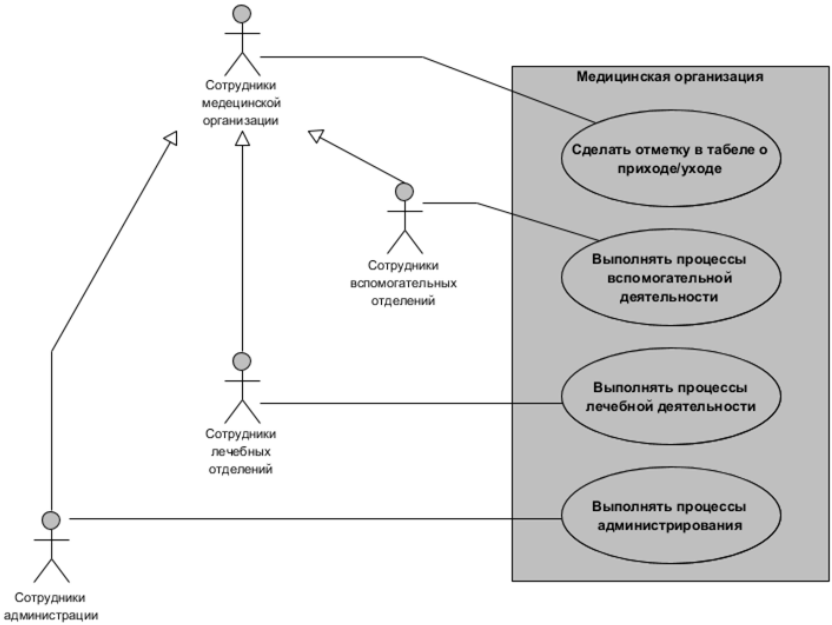


Рисунок 7 – Пример модели типа «Вариант использования» для гипотетической медицинской организации

В продолжение подхода, который использовался для модели структуры (см. Рисунок 3), опишем модель на естественном языке – в конце концов, мы учимся «читать и писать», используя UML. Итак, смотрим на Рисунок 7.

1. В медицинской организации все «Сотрудники медицинской организации» в границах структуры «Медицинская организация» должны выполнить ИБП «Сделать отметку в табеле приходе/уходе»;

2. Если сотрудник входит в группу «Сотрудники администрации», то он должен выполнять процессы «Выполнять процессы администрирования»;

3. Если сотрудник входит в группу «Сотрудники лечебных отделений», то он должен выполнять процессы «Выполнять процессы лечебной деятельности»;

4. Если сотрудник входит в группу «Сотрудник вспомогательных отделений», то он должен выполнять процессы «Выполнять процессы вспомогательной деятельности».

В действительности здесь содержатся три сценария или варианта использования процессов, существующих в организации:

- **Вариант 1.** Выполнение п.п. 1 и 2;
- **Вариант 2.** Выполнение п.п. 1 и 3;
- **Вариант 3.** Выполнение п.п. 1 и 4.

Надеюсь, что этот пример убедительно показал, почему данный тип моделей UML называется «Модель вариантов использования».

И еще, надеюсь, понятно почему модели UCD относят к группе поведенческий моделей: в этих моделях присутствует такой компонент, как «Исполнитель», а с каждым типом исполнителей связывается свой сценарий поведения (вариант использования) или, как говорит известная русская пословица – «Что может поп – того не может пономарь»[6] …

Однако, если для специалиста, работающего в области медицины (как, впрочем, и любой другой области), понятно для чего нужна его организация, то для стороннего субъекта может быть не понятно для чего нужны процессы в данной организации. В целях разъяснения того, для чего нужна та или иная группа процессов в нотации UML существует еще один элемент – «Взаимодействие» (см. Рисунок 8), который инкапсулирует описание требований к выполнению процессов, которые с этим элементом связаны, и в явном виде связывается с элементами UCD, которые должны взаимодействовать для решения задачи взаимодействия.

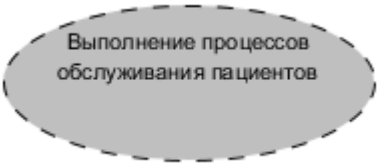


Рисунок 8 – Элемент нотации UML «Взаимодействие»

[6] А, кстати, чем не модель верхнего уровня для священнослужителей: имеется два типа исполнителей: «Поп» и «Пономарь»; «Поп» выполняет свои процессы или «Выполнение процессов Попа», а «Пономарь» выполняет свои процессы или «Выполнение процессов Пономаря».

В соответствии со спецификацией OMG (см. [3]), *элемент «Взаимодействие» описывает структуру взаимодействующих элементов (или ролей), каждый из которых выполняет свою специализированную функцию, но совместно эти элементы обеспечивают некоторую желаемую функциональность. Основное назначение элемента «Взаимодействие» состоит в том, чтобы пояснить, как работает система, и, следовательно, этот элемент обычно включает в себя только те аспекты реальности, которые считаются относящимися к пояснению. Таким образом, такие детали, как идентичность или точный класс фактических участвующих экземпляров, скрываются.*

Один из элементов нотации, который может связывать элемент «Взаимодействие» с другими элементами модели, это уже рассмотренный элемент «Ассоциация» (см. Рисунок 6). При его использовании предполагается, что между элементами модели происходит обмен информацией по типу облачных технологий, т. е. можно предположить, что элемент «Взаимодействие» выступает в качестве облака или черного ящика, который может принимать информацию, обрабатывать принятую информацию (но механизм обработки инкапсулируется) и передавать результаты обработки другим элементам, которые участвуют в некотором глобальном процессе, и, которые его обеспечивают.

Примером может служить модель, показанная выше (см. Рисунок 7), но дополненная элементом «Взаимодействие» (см. ниже Рисунок 9).

Если интерпретировать эту новую модель на естественном языке, например, для сценария «Если сотрудник входит в группу «Сотрудники лечебных отделений», то он должен выполнять процессы «Выполнять процессы лечебной деятельности»», который использовался при интерпретации к предыдущей модели (см. Рисунок 7), то новая интерпретация может выглядеть так: **«Если сотрудник входит в группу «Сотрудники лечебных отделений», то он должен выполнять процессы «Выполнять процессы лечебной деятельности», которые обеспечивают глобальный процесс «Выполнение процессов обслуживания пациентов»».**

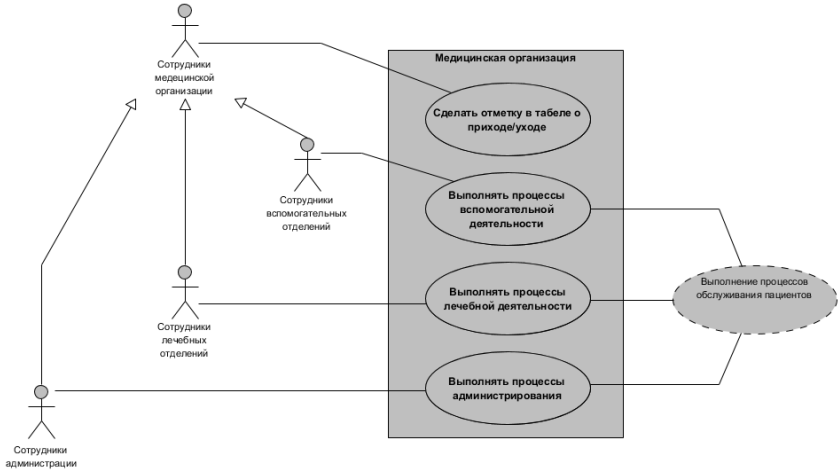


Рисунок 9 – Пример модели типа «Вариант использования» для гипотетической медицинской организации, дополненной элементом «Взаимодействие»

Допустим, что мы хотим детализировать и выделить некоторые процессы, которые, например, выполняет сотрудник лечебного отделения. Пусть это будут процессы: «Постановка диагноза», «Назначение лечения», «Назначение диеты». Т. е. эти процессы входят в вариант использования «Выполнять процессы лечебной деятельности». В нотации UML для этих целей используется элемент «Зависимость», который в границах модели «Вариант использования» имеет стереотип «Включение» и отображается, как показано на рисунке ниже (см. Рисунок 10). Для упрощения изложения, в дальнейшем будем говорить о элементе «Включение» (или «include»).

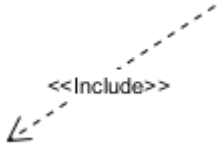


Рисунок 10 – Элемент «Зависимость» с используемым стереотипом «Включение»

Элемент «Включение» является направленной связью между двумя элементами «Вариант использования» и подразумевает, что элемент «Вариант использования», от которого направлена стрелка», использует поведение элемента «Вариант использования», к которому направлена стрелка. Элемент «Включение» может быть именован для лучшего понимания поведения включаемого элемента в контексте включающего варианта использования.

Некоторые инструменты разработчика (например, Visual Paradigm) позволяют редактировать наименование стереотипа и указывать его значение по-русски для лучшего понимания модели русскоязычным пользователем.

С учетом вышесказанного, мы можем расширить модель, показанную выше, (см. Рисунок 9), элементом «Включение» (см. Рисунок 11).

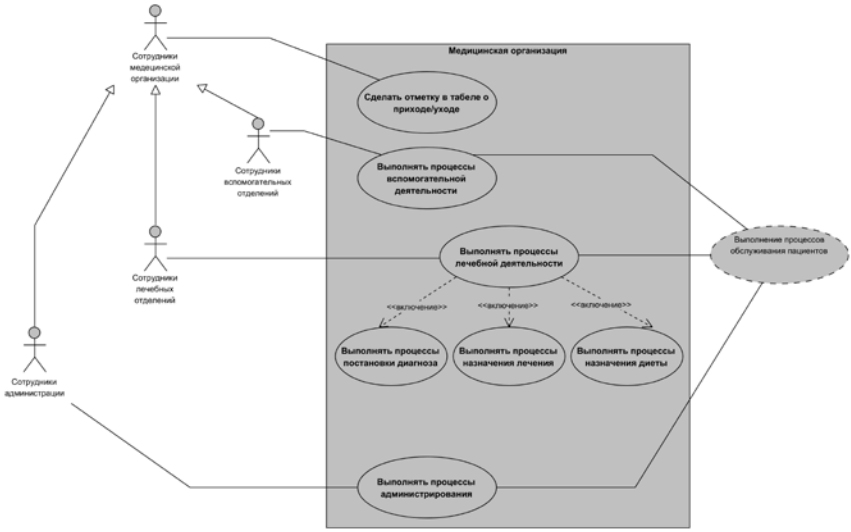


Рисунок 11 – Пример модели типа «Вариант использования» для гипотетической медицинской организации, дополненной элементом «Включение» и связанными с ним вариантами использования

Если интерпретировать эту новую модель на естественном языке, например, для сценария «Если сотрудник входит

в группу «Сотрудники лечебных отделений», то он должен выполнять процессы «Выполнять процессы лечебной деятельности», которые обеспечивают глобальный процесс «Выполнение процессов обслуживания пациентов»», который использовался при интерпретации к предыдущей модели (см. Рисунок 9), то новая интерпретация может выглядеть так: «Если сотрудник входит в группу «Сотрудники лечебных отделений», то он должен выполнять процессы «Выполнять процессы лечебной деятельности», включающие процессы «Выполнять процессы постановки диагноза», «Выполнять процессы назначения лечения», «Выполнять процессы назначения диеты», которые обеспечивают глобальный процесс «Выполнение процессов обслуживания пациентов»».

Чем больше примеров мы рассматриваем и вводим новые элементы, тем понятнее становится необходимость использования нотации UML – у автора возникает стойкая ассоциация, что появление нотации UML сродни появлению современной нотации в математике: если в «Началах» Евклида все излагалось словесно, то в работах Ф. Виета, Г. В. Лейбница и др. появляется современная компактная запись математических формул. Спасибо им! За UML скажем спасибо Г. Бучу, Д. Рамбо, И. Якобсону [2].

Однако, продолжим наше погружение в модели «Вариант использования».

Возможна следующая ситуация, при которой существует некоторый процесс, общий для нескольких вариантов использования. Возвращаясь к нашему примеру, это может быть, например, процесс подготовки отчетов для нескольких исполнителей. Как это отобразить в модели?

В нотации UML для этих целей используется элемент «Зависимость», который в границах модели «Вариант использования» имеет стереотип «Расширение» и отображается, как показано на рисунке ниже (см. Рисунок 12). Для упрощения изложения, в дальнейшем будем говорить о элементе «Расширение» (или «Extend»).

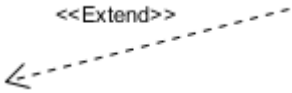


Рисунок 12 – Элемент «Зависимость»
с используемым стереотипом «Расширение»

Это отношение указывает, что поведение варианта использования может быть расширено за счет другого (обычно дополнительного) варианта использования. Один и тот же дополнительный вариант использования может расширять несколько вариантов использования.

С учетом вышесказанного, мы можем расширить модель, показанную выше, (см. Рисунок 11), элементом «Расширение» (см. Рисунок 13).

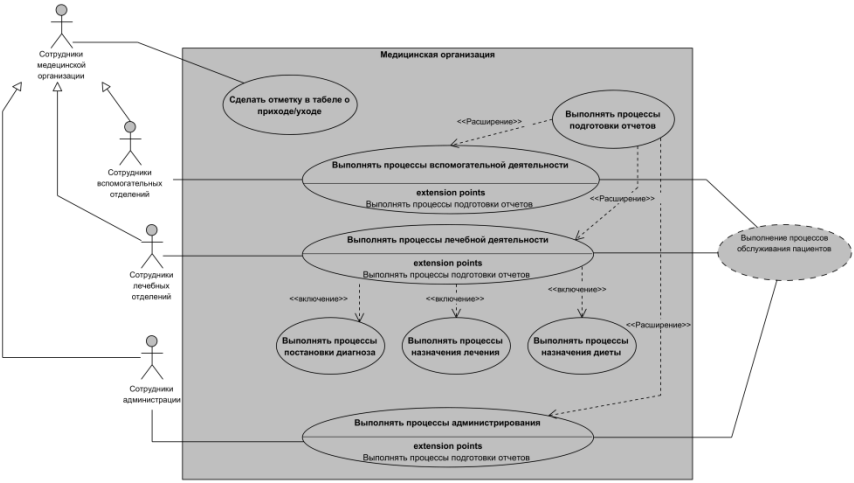


Рисунок 13 – Пример модели типа «Вариант использования» для гипотетической медицинской организации, дополненной элементом «Расширение» и связанными с ним вариантами использования

Обратите внимание, что несколько изменился вид элементов «Вариант использования» – элемент разделился на две части, нижняя из которых получила наименование «extension points» (точки расширения) с указанием наименования

расширяющего сценария (в нашем примере это сценарий «Выполнять процессы подготовки отчетов»). Точек расширения может быть несколько, а, следовательно, список точек расширения будет увеличен.

Т. е. новая интерпретация модели на естественном языке может быть такой: «Если сотрудник входит в группу «Сотрудники лечебных отделений», то он должен выполнять процессы «Выполнять процессы лечебной деятельности», включающие процессы «Выполнять процессы постановки диагноза», «Выполнять процессы назначения лечения», «Выполнять процессы назначения диеты», а также расширенные процессом «Выполнять процессы подготовки отчетов», которые обеспечивают глобальный процесс «Выполнение процессов обслуживания пациентов»».

Рассмотрим еще один важный элемент, необходимость которого можно пояснить следующим примером. На рисунке выше (см. Рисунок 13) показано, что, в частности, исполнители «Сотрудники лечебных отделений» могут выполнять сценарии «Выполнять процессы постановки диагноза», «Выполнять процессы назначения лечения», «Выполнять процессы назначения диеты». Но медицина достаточно регламентированная деятельность, которая должна соответствовать требованиям нормативно-справочных документов, начиная от законодательства и приказов Минздрава РФ, и, заканчивая внутренними приказами и распоряжениями в медицинской организации. Т. е., перечисленные выше, сценарии зависят от нормативно-справочных документов. Для отображения такой ситуации в нотации UCD существует элемент «Зависимость», который показан ниже (см. Рисунок 14).

Рисунок 14 – Элемент «Зависимость»

В отличие от элементов «Расширение» и «Включение», рассмотренных выше, элемент «Зависимость» не именуется и показывает, что один элемент влияет на работу другого (стрелка направляется к управляющему элементу). Принято называть управляющий сценарий «Поставщиком», а зависимый сценарий – «Клиентом».

С учетом сказанного, внесем дополнение в модель, показанную выше (см. Рисунок 13), и покажем зависимость варианта использования «Выполнять процессы лечебной деятельности» от варианта использования «Использовать нормативно – справочные документы» (см. Рисунок 15).

Интерпретировать эту модель можно, например, для варианта использования «Выполнять процессы постановки диагноза» следующим образом: «Сотрудники лечебных отделений могут выполнять процессы лечебной деятельности, как, например, выполнять процессы постановки диагноза, но при этом использовать нормативно-справочные документы».

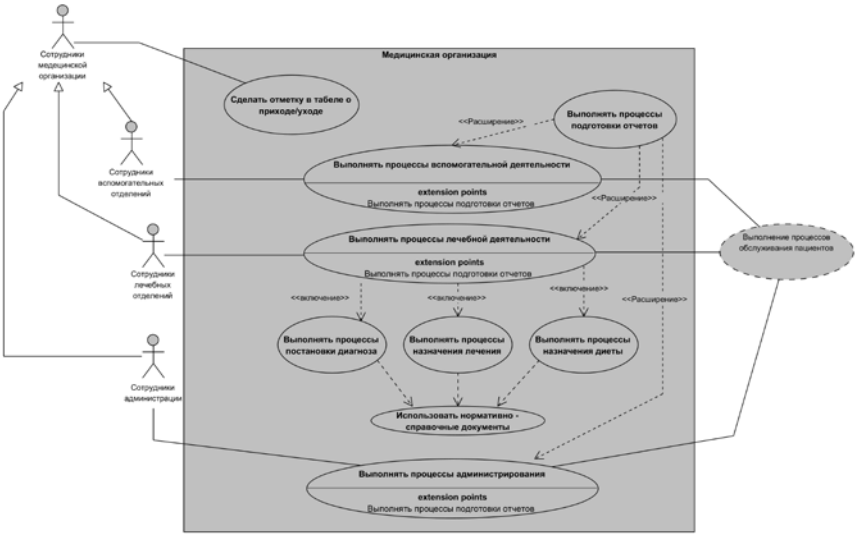


Рисунок 15 – Пример модели типа «Вариант использования» для гипотетической медицинской организации, дополненной элементом «Зависимость» и связанными с ним вариантами использования

При моделировании вариантов использования часто возникает ситуация, при которой некоторый вариант использования может быть выполнен только при выполнении определенного условия. Например, невозможно использовать сценарий «Выполнять процессы назначения лечения» до того, как выполнен сценарий «Выполнять процессы постановки диагноза». Для моделирования подобных ситуаций существует элемент «Ограничение», который показан ниже (см. Рисунок 16).

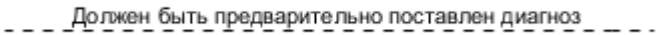


Рисунок 16 – Пример элемента «Ограничение»

Элемент «Ограничение» вставляют между связями с указанием ограничения.

В завершении раздела следует отметить, что часто возникает необходимость комментариев к тем элементам, которые использованы в модели. Для этих целей используется элемент «Примечание», который показан ниже (см. Рисунок 17).

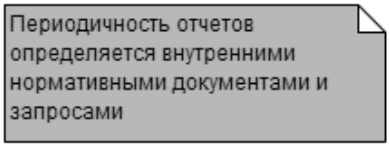


Рисунок 17 – Пример элемента «Примечание»

Элемент «Примечание» соединяется с элементом, который он комментирует посредством пунктирной линии.

Пример использования элементов «Ограничение» и «Примечание» показан ниже (см. Рисунок 18).

Однако использование UCD позволяет увидеть варианты использования, но последовательность их исполнения в этом типе моделей UML не позволяет однозначно показать последовательность выполнения вариантов использования. Модели, которые позволяют однозначно определить последователь-

ность выполнения сценариев, описываются в следующем разделе.

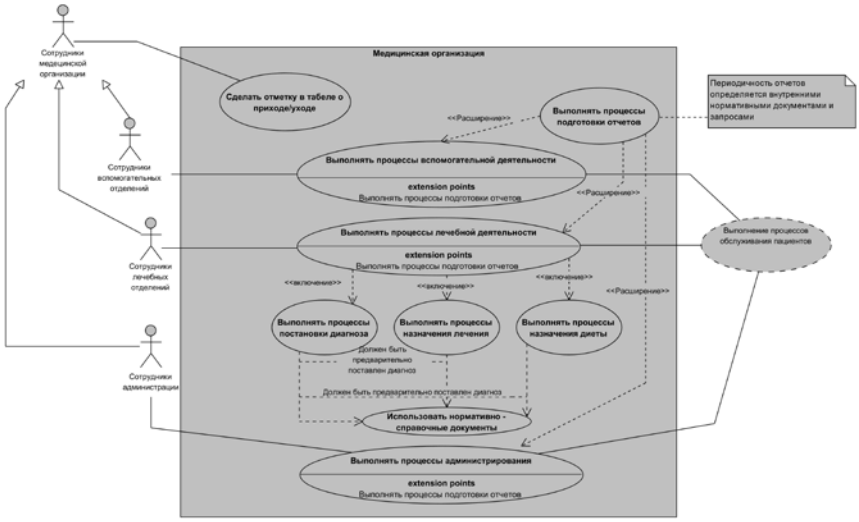


Рисунок 18 – Пример использования элементов «Ограничение» и «Примечание»

C. Модель «Последовательность» (Sequence Diagram)

Выше (см. раздел «ГЛАВА 1.b. МОДЕЛЬ «ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ» (USE CASE DIAGRAM)») была рассмотрена модель, которая отвечает на вопросы Диаграммы Захмана «КАК», «КТО», «ГДЕ», но не отвечает на остальные вопросы: «ЧТО», «ПОЧЕМУ», «КОГДА».

Ниже рассмотрим модель типа «Последовательность» (далее – SeqD), которая может частично ответить на вопросы «ЧТО» (если в модели используется элемент, ссылающийся на данные) и «КОГДА», а также расширяет ответы на вопросы «КАК», «КТО» и «ГДЕ».

В чем польза от применения SeqD? Если в UCD показаны участники и сценарии использования, но не очевидна последовательность выполнения сценариев, то SeqD показывает в какой

последовательности и кем или чем должны выполняться сценарии, а также временные интервалы, необходимые для каждого из сценариев[7].

О SeqD также можно сказать, что это своеобразный «язык», который имеет свои «подлежащие», «сказуемые» и др. элементы естественных языков. Но в отличие от UCD, SeqD имеет большее количество элементов моделирования и, частично, может использовать элементы из UCD.

Одним из основных элементов в SeqD является элемент, который в литературе принято называть «Линия жизни» (от английского «LifeLine»), и, который именуется либо существительным, либо именем собственным, а, по существу, в нотации SeqD выполняет роль подлежащего, т. е. отвечает на вопрос диаграммы Захмана «КТО?». В общем случае пример графического представления элемента «Линия жизни» показан ниже (см. Рисунок 19). Если использовать парадигму с естественным языком, описанную выше, то, глядя на Рисунок 19, можно было бы сказать: «Медицинская организация что-то делает». А, что она делает? Давайте рассмотрим далее, КАК получить развернутый ответ.

Рисунок 19 – Пример элемента «Линия жизни»

Согласно спецификации UML (см. [3]), «Линия жизни» представляет в модели SeqD отдельного участника..

[7] Впрочем, временные интервалы могут и не указываться, если они не существенны для описываемых процессов.

Но не следует забывать, что под «Участником» в большинстве нотаций (включая UML) понимают некий объект, который участвует в выполнении ИБП. И не важна сущность объекта – он одушевленный или нет – главное, что объект участвует в выполнении некоторого процесса.

Пунктирная линия элемента «Линия жизни» представляет собой некоторую ось времени, перемещаясь по которой (ось направлена сверху вниз), можно отмечать начало и завершение некоторых событий. Событиями для SeqD будет являться обмен сообщениями, которые отправляются от одного участника к другому, но возможен вариант, при котором участник отправляет сообщение самому себе.

Обмен сообщениями отображается с помощью стрелок (элемент «Сообщение» или, как в оригинальной спецификации UML (см. [3]), «Message»), указывающих направление сообщения, и тонких прямоугольных элементов (элемент «Спецификация выполнения», или, как в оригинальной спецификации UML, «ExecutionSpecification»).

На рисунке ниже (см. Рисунок 20) показан пример обмена сообщениями типа «Сообщение-вызов» и «Сообщение-ответ».

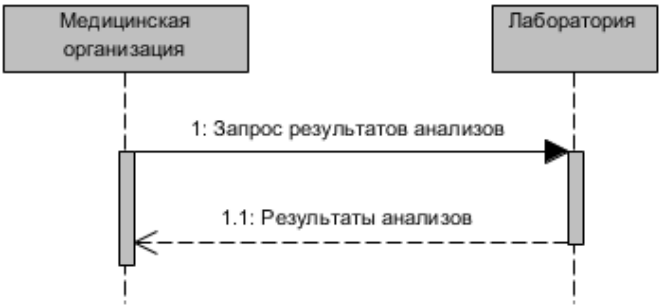


Рисунок 20 – Пример обмена сообщениями между элементами «Линия жизни»

Сообщение-вызов (как в данном примере «Запрос результатов анализов») представляет собой стрелку со сплошной линией, которая связывает две линии жизни и направлена

перпендикулярно к ним, т. е. от линии жизни участника инициатора обмена сообщениями к линии жизни участника, который принимает сообщение (в данном примере это «Лаборатория»). При этом, на линии жизни «Медицинская организация» появился элемент «Спецификация выполнения», из верхней части которого исходит стрелка, а на линии жизни «Лаборатория» также появился элемента «Спецификация выполнения», в верхнею часть которого стрелка входит. Это трактуется следующим образом: в медицинской организации был запущен процесс, который предполагает запрос анализов в лаборатории, что предполагает запуск процессов в лаборатории по подготовке ответа в виде сообщения «Результаты анализов», который отправляется обратно к инициатору диалога. На рисунке это соответствует стрелке с пунктирной линией (элемент «Сообщение-ответ»), направленной от запрошенного участника к участнику-инициатору. При этом, элементы типа «Спецификация выполнения» ограничиваются (закрываются) снизу. Диалог завершен.

Рисунок 21 – Пример элемента «Исполнитель»

Особое место в отображении элементов «Линия жизни» занимает элемент (по сути, специфическая линия жизни) «Исполнитель», пиктограмма которого такая же, как у элемента «Исполнитель» в UCD, и смысл тот же, но внизу добавляется элемент «Спецификация выполнения» по всей длине линии жизни типа «Исполнитель» без разрывов[8] (см. Рисунок 21).

Во многих инструментах моделирования автоматически расставляются номера, которые показывают последовательность обмена сообщениями и их принадлежность к конкретному диалогу.

[8] Очень оптимистично – пока идут процессы в системе, с исполнителем ничего плохого не будет. Вот – нужно делать системы, которые работают, работают и работают….

В примере, приведенном выше, около именований стрелок стоят номера:

- «1: Запрос результатов анализов» показывает, что именно это сообщение является инициатором диалога;
- «1.1: Результаты анализов» показывает, что это сообщение относится к группе сообщений, активированных сообщением 1, и будет выполнятся вторым в группе.

Элементов типа «Сообщение» в нотации SeqD несколько видов. Рассмотрим эти виды ниже.

i. Элемент «Инициирующее сообщение»

Элемент «Инициирующее сообщение», согласно спецификации UML (см. [2] и [3], а также Приложение 1. Элементы нотации UML), конкретизирует связь между взаимодействующими линиями жизни и вызывает выполнение операции в целевой линии жизни.

Это сообщение, как, впрочем, и другие сообщения в SeqD, не может существовать в модели сам по себе, т. е. необходимо наличие двух линий жизни, которые связаны посредством инициирующего события: одна линия жизни является источником информации, а вторая – приемником или, как пишут в оригинальной литературе, мишенью. В соответствии с нотацией SeqD, все сообщения нумеруются, что и понятно: если изобразить в модели множество сообщений без нумерации, то будет трудно понять какова последовательность взаимодействия между линиями жизни. Нотация UML рекомендует, но не требует именовать сообщения, т. к. в этом случае повышается читабельность модели – становится понятным, ЧТО именно передается из одной линии жизни в другую. Автор твердый сторонник этой рекомендации – ИМЕНОВАТЬ!

Рассмотрим примеры использования элемента «Инициирующее сообщение».

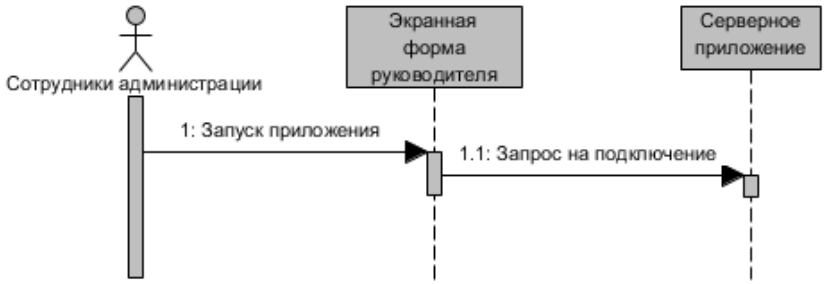


Рисунок 22 – Пример использования элемента «Инициирующее сообщение»

В примере выше (см. Рисунок 22) показан сценарий, в котором отмечено, что кто-либо из группы пользователей «Сотрудники администрации» запускает на своем рабочем месте некоторое приложение (т. е. видит экранную форму) (см. цепочку: *линия жизни «Сотрудники администрации»* → *инициирующее сообщение «Запуск приложения»* → *линия жизни «Экранная форма руководителя»*), и, которое запускает, в свою очередь, серверное приложение (см. цепочку: *линия жизни «Экранная форма руководителя»* → *инициирующее сообщение «Запрос на подключение»* → *линия жизни «Серверное приложение»*).

Будем оставаться верными парадигме, которую установили еще в Томе I настоящего учебного пособия: мы «читаем» модели. В данном случае, модель, которую иллюстрирует Рисунок 22, может быть «прочитана» следующим образом:

«*Некто, входящий в группу пользователей «Сотрудники администрации», запускает приложение и видит экранную форму, которая после загрузки (на что требуется некоторое время) отправляет запрос на подключение соответствующему серверному приложению*».

Но то, что описано, из серии «Вопрос повис в воздухе», т. е. «запустили», «отправили», а, что дальше?

Чтобы не было «висящих» вопросов, рассмотрим следующий элемент SeqD.

ii. Элемент «Возвращаемое сообщение»

Элемент «Возвращаемое сообщение», согласно спецификации UML (см. [2] и [3], а также Приложение 1. Элементы нотации UML), конкретизирует связь между взаимодействующими линиями жизни и соответствует передаче информации от вызываемой линии жизни к вызывающей линии жизни в соответствии с инициирующим сообщением

Этот элемент во многом похож на элемент «Инициирующее сообщение», т. е. должны быть источник информации и мишень, сообщение нумеруется и именуется, но отличие состоит в том, что возвращаемое сообщение может существовать только в паре с инициирующим сообщением и возникает только в том случае, если инициирующее сообщение предполагает отклик. Это отличие подчеркивается тем, что возвращаемое сообщение отображается в виде пунктирной стрелки.

Продолжим пример, который рассматривался выше (см. Рисунок 22). Дополним модель возвращаемыми сообщениями (см. Рисунок 23).

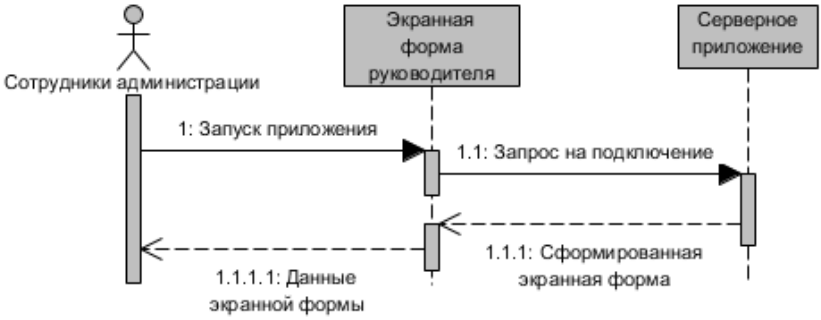


Рисунок 23 – Пример использования элемента «Возвращаемое сообщение»

В данном случае, модель, которую иллюстрирует Рисунок 23 может быть «прочитана» следующим образом:

«*Некто, входящий в группу пользователей «Сотрудники администрации», запускает приложение и видит экранную форму, которая после загрузки (на что требуется некоторое время) отправляет запрос на подключение соответствующему серверному приложению. Серверное*

приложение отправляет сформированную экранную форму на рабочий компьютер сотрудника администрации в линию жизни «Экранная форма руководителя», а данные экранной формы может видеть член группы «Сотрудники администрации», который запускал приложение».

Во многих случаях можно обойтись таким представлением последовательности выполняемых действий, но, если нужна дополнительная информация о том, что происходит внутри некоторой линии жизни, например, к каким подсистемам происходят обращения через отправку сообщений, то нужна дополнительная информация об этом.

iii. Элемент «Внутреннее сообщение»

Внутреннее сообщение – это сообщение, которое адресуется внутри одной линии жизни, т. е. в границах одной линии жизни некоторая операция отправляет сообщение к другой операции, которая относится к этой же линии жизни

Внутреннее сообщение, с одной стороны, показывает, что в границах одной линии жизни отправлено сообщение, которое должно инициировать процесс выполнения инициирующего сообщения в соответствии с его содержанием, а, с другой стороны, внутреннее сообщение через его именование отображает, ЧТО необходимо сделать при обработке инициирующего сообщения. Размеры внутреннего сообщения показывают относительную продолжительность его обработки.

Демонстрацией этого пассажа может служить пример (см. ниже Рисунок 24), продолжающий примеры, показанные выше (см. Рисунок 22 и Рисунок 23).

Эта модель может быть «прочитана» путем дополнения содержания интерпретаций, приведенных выше (см. п.п. ГЛАВА 1.c.i и ГЛАВА 1.c.ii):

«*Некто, входящий в группу пользователей «Сотрудники администрации», запускает приложение и видит экранную форму, которая после загрузки (на что требуется некоторое время, связанное с процессом «Обработать сообщение «Запуск приложения») отправляет запрос на подключение соответствующему серверному приложению. Серверное приложение обрабатывает пришедшее сообщение (на что требуется некоторое время, связанное с процессом «Обработать сообщение «Запрос на подключение») и*

отправляет сформированную экранную форму на рабочий компьютер сотрудника администрации в линию жизни «Экранная форма руководителя», в которой происходить отображение сформированной экранной формы, а данные экранной формы может видеть член группы «Сотрудники администрации», который запускал приложение».

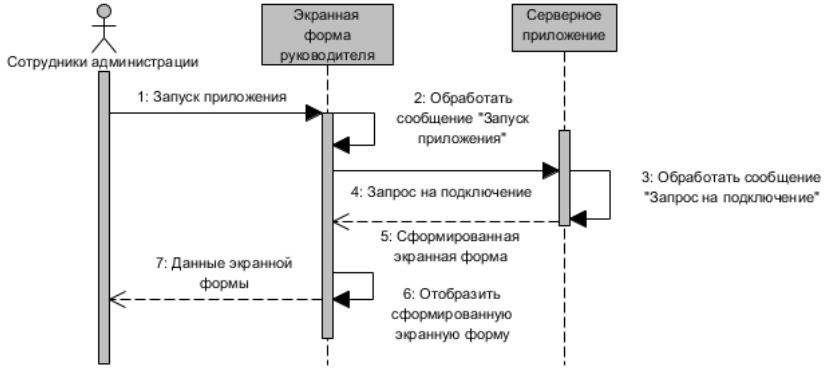


Рисунок 24 – Пример использования элемента «Внутреннее сообщение»

Наше описание увеличивается в размерах… Но, с другой стороны, графика, которая сопровождает описание элементов, наглядная демонстрация известной истины: «Лучше один раз увидеть, чем 100 раз услышать».

iv. Элемент «Рекурсивное сообщение»

Рекурсивное сообщение – это сообщение, которое адресуется внутри одной линии жизни в рамках одного процесса, но таким образом, что в границах этого процесса создается дополнительный процесс.

Рассмотрим использование элемента «Рекурсивное сообщение» на следующем примере.

Пусть в некоторой торговой компании существует следующая практика распределения поступающего товара по трем складам:

1. Сотрудник администрации направляет на склад 1 товар в количестве 50% от поступившего;

2. Сотрудник администрации направляет на склад 2 товар в количестве 50% от оставшегося после отправки на склад 1;

3. Сотрудник администрации направляет на склад 3 товар в количестве 50% от оставшегося после отправки на склад 3.

Вдумчивый читатель может спросить: а как быть, если общее количество поступившего товара составляет 1 единицу (например, чашек, телевизоров, …)?

По сути, это будет некоторый нетипичный случай или на ИТ-терминологии – исключение. Работу с исключениями в SeqD рассмотрим ниже (см. ГЛАВА 1.c.viii.5 и ГЛАВА 1.c.viii.6).

Пусть существует некоторое программное обеспечение, которое содержит в своем составе функцию, решающую задачу, приведенную выше. Модель SeqD верхнего уровня, отображающая работу функции показана ниже (см. Рисунок 25).

Рисунок 25 – Пример использования элемента «Рекурсивное сообщение»

Продолжим нашу традицию – «прочтем» модель.

«*Некто, входящий в группу пользователей «Сотрудники администрации», видит экранную форму «Экранная форма руководителя» и направляет к ней запрос «Запрос определения количества товара, отправ-*

ленного на склады». Этот запрос экранная форма интерпретирует в запрос «Запрос к функции расчета количества товара на складах», который направляется на «Серверное приложение». Функция, входящая в состав серверного приложения, по алгоритму, приведенному выше, вычисляет количество товара для отправки на склад 1, результат отправляет для визуализации в экранную форму руководителя, а остаток направляет, обращаясь сама к себе (т. е. рекурсивно) для вычисления количества товара для отправки на склад 2. Процесс повторяется до момента завершения вычисления количества товара, отправляемого на склад 1, склад 2, склад 3. Остаток товара, оставшийся нераспределенным, также записывается в экранную форму через возвращаемое сообщение «Запись в экранную форму остатков». Результаты отображения возвращаются к пользователю через возвращаемое сообщение «Результаты запроса»».

Уф… Написал. Оставляю на усмотрение читателя принятие решения: что ему проще – писать длиннющие тексты, которые очень неохотно читают (программисты уж точно – и читать не будут, а сделают «по-понятиям»), или рисовать модели в стандартном представлении, на основании которых можно и строго спросить: «я тебе, ЧТО изобразил, а, ЧТО ты сделал?». Или поблагодарить программиста за отлично сделанную работу.

Это всего лишь сценарии возможного развития производственных отношений между аналитиком и его коллегами. Ничего личного.

v. Элемент «Порождающее сообщение»

Порождающее сообщение – это сообщение предназначено для формирования новой линии жизни, которой не было в исходном состоянии некоторой системы.

Предположим, что существует некоторая система, на которой выполняется серверное приложение. Это серверное приложение может, при необходимости, обращаться динамически к библиотеке функций, которые подгружаются, выполняются и возвращают результаты в вызывающее приложение. На практике это хорошо известные программистам динамически присоединяемые библиотеки (Dynamic Link Library или DLL).

Пример отображения такой ситуации показан ниже (см. Рисунок 26).

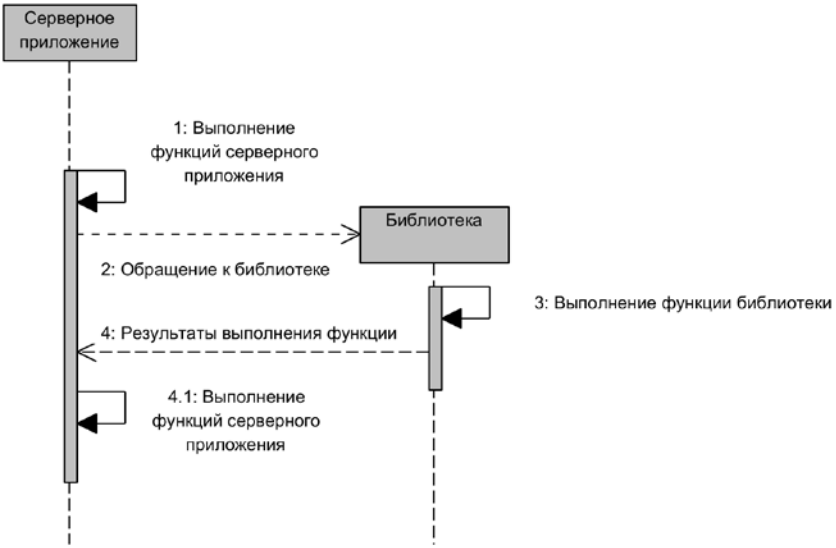


Рисунок 26 – Пример использования элемента «Порождающее сообщение»

Текстовая интерпретация:

«Некоторое серверное приложение выполняет свои функции (см. внутренний поток «Выполнение функций серверного приложения»), но в некоторый момент времени требуется подгрузка функций, хранимых во внешней библиотеке. Это осуществляется через поток «Обращение к библиотеке», который активирует линию жизни «Библиотека», что приводит к появлению внутреннего потока «Выполнение функции библиотеки». По завершению выполнения функции библиотеки, результаты возвращаются в вызывавшее серверное приложение, которое продолжает функционировать».

Читатель может задать вопрос: «А, если подгруженная библиотека больше не понадобится, то она будет находиться в памяти совершенно бесполезно и просто занимать место?». На практике хорошо известно, как исправить такую ситуацию: просто выгрузить, ставшею ненужной, библиотеку и освобо-

дить место в памяти. Как изобразить это в SeqD? В нотации SeqD это предусмотрено (см. ГЛАВА 1.c.vi).

vi. Элемент «Уничтожающее сообщение»

Уничтожающее сообщение – это сообщение, которое приведет к завершению целевой линии жизни.

Отображение ответа, который был поставлен в конце предыдущего раздела, показано ниже (см. Рисунок 27).

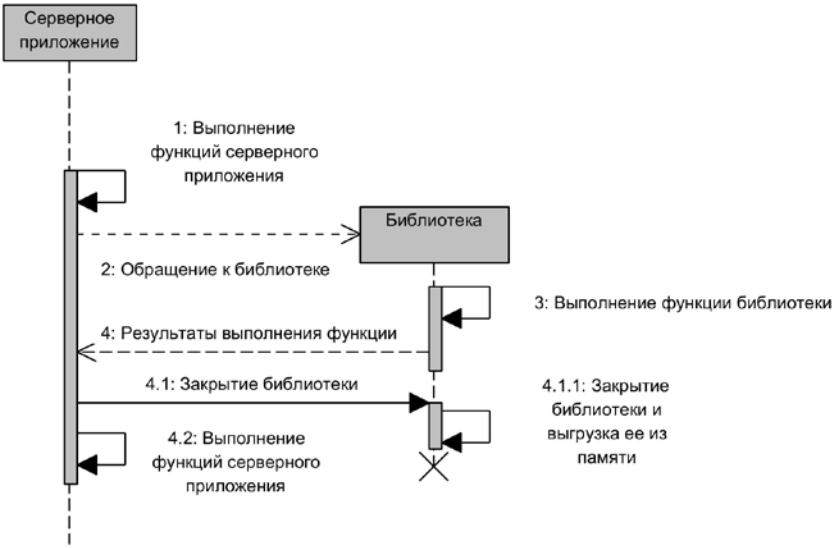


Рисунок 27 – Пример использования элемента «Уничтожающее сообщение»

Текстовая интерпретация:

«Некоторое серверное приложение выполняет свои функции (см. внутренний поток «Выполнение функций серверного приложения»), но в некоторый момент времени требуется подгрузка функций, хранимых во внешней библиотеке. Это осуществляется через поток «Обращение к библиотеке», который активирует линию жизни «Библиотека», что приводит к появлению внутреннего потока «Выполнение функции библиотеки». По завершению выполнения функции библиотеки, результаты

возвращаются в вызывавшее серверное приложение, которое инициирует закрытие библиотеки через поток «Закрытие библиотеки». Само серверное приложение продолжает работать».

Обратите внимание на символ, которым завершается линия жизни «Библиотека», т. е. ☒.

Таким образом в нотации SeqD отображается завершение линии жизни или, если рассмотреть реальную ситуацию с приложением, выгрузка библиотеки из памяти.

vii. Элемент «Длительное сообщение»

Уничтожающее сообщение – это сообщение, которое показывает в явном виде, что от момента отправки сообщения к целевой линии жизни до начала выполнения некоторого процесса в принимающей линии жизни проходит никоторое время

Действительно, при моделировании реальных процессов, в которых между событиями проходит некоторое время (часто весьма значительное, которым невозможно пренебречь), необходимо это отразить в модели SeqD. Например, между отправкой товара и поступлением его на склад может пройти и сутки, и двое, и более. При отправке товара готовятся сопровождающие документы, которые из некоторой информационной системы поставщика попадут в информационную систему на складе в течение нескольких секунд (как правило). Но товар еще не пришел. Т. е. поставить отметки в электронном документе, подтверждающие получение товара, возможно будет только тогда, когда придет товар. Имеет смысл задержать отправку документов, приурочив это к моменту физической доставки товара. Модель этой ситуации показана ниже (см. ниже Рисунок 28).

Текстовая интерпретация:

«*В информационной системе поставщика происходит подготовка сопровождающих документов, но в информационную систему склада эти документы отправляются с задержкой в 2 дня. После прихода сопровождающих документов, информационная система склада отправляет подтверждающее сообщение в информационную систему поставщика*».

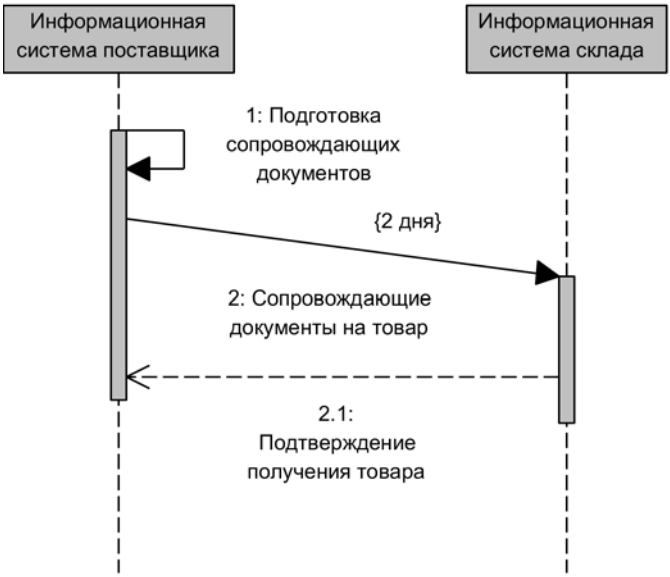


Рисунок 28 – Пример использования элемента «Длительное сообщение»

Обратите внимание, что в элементе «Длительное сообщение» появился, кроме текстового описания элемента, дополнительный элемент в фигурных скобках, который показывает длительность сообщения {2 дня}.

viii. Элементы «Фрагменты последовательностей»

Казалось бы, что описаны все возможные элементы, которые нужны для моделирования последовательности действий при выполнении некоторого процесса: линии жизни, разные типы сообщений. Но, жизнь гораздо сложнее, чем этого хотелось бы каждому из нас, включая аналитиков. Как быть, если в некоторых случаях процесс может пойти по-иному или вообще недопустим, или может выполняться ограниченное число раз, или …?

К счастью, разработчики нотации UML учли реалии жизни и далее мы рассмотрим следующую группу элементов,

входящих в SeqD, и, называемых «Фрагменты последовательностей» (*англ. Sequence Fragments*).

1. Альтернативные фрагменты последовательностей (англ. Alternative Sequence Fragments)

Альтернативные фрагменты последовательностей – это элемент, который может охватывать несколько линий жизни, включая спецификации выполнения и сообщения между ними, таким образом, чтобы показать альтернативные варианты для спецификаций и сообщений, т. е. запущена будет только та последовательность, для которой выполняется определенное условие

Типовой пример альтернативного фрагмента последовательностей показан ниже (см. Рисунок 29).

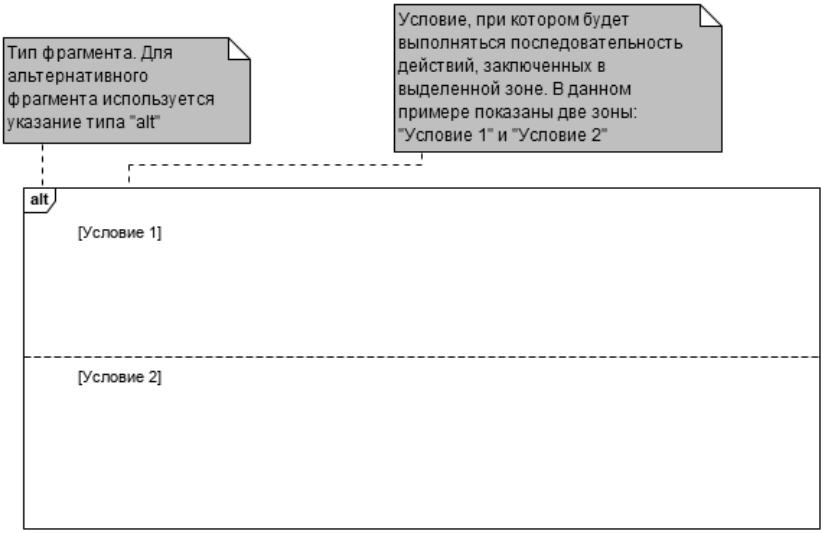


Рисунок 29 – Альтернативный фрагмент последовательностей

Следует отметить, что все фрагменты последовательностей имеют общий элемент в верхнем левом углу, который для альтернативного фрагмента имеет значение «alt» (alt). Тело альтернативного фрагмента разбивается на горизонтальные зоны, которые именуются условием выполнения последовательности действий внутри зоны (например, «Если время ожидания > 5 се-

кунд» или, как показано на рисунке [Условие 1]). Количество зон не может быть меньше 1. Если количество зон равно 1, то используется вырожденный вариант альтернативного фрагмента, который в SeqD имеет отдельный тип, называемый *Возможный фрагмент последовательностей* (см. ГЛАВА 1.c.viii.2). Максимальное количество определяется, главным образом, возможностью размещения модели на листе формата А4 или А3 так, чтобы надписи и другие значимые элементы читались. Возможно, что в вашей организации действуют какие-либо корпоративные стандарты на формат рисунков, например, требования использования форматов А1 или А2.

Рассмотрим применение альтернативного фрагмента последовательностей на конкретном примере.

Пусть существует некоторая система, в которой функционируют два серверных приложения: «Серверное приложение 1» и «Серверное приложение 2». На серверном приложении 1 выполняется функция 1, которая завершается подготовкой данных двух видов: вида 1 и вида 2. Эти данные передаются на обработку в серверное приложение 2 таким образом, что, если подготовлены данные вида 1, то должна быть запущена функция 2, а, если подготовлены данные вида 2, то должна быть запущена функция 3. Каждая из функций возвращает результаты своей работы в серверное приложение 1. Модель этого процесса (с использованием альтернативного фрагмента) показана ниже (см. Рисунок 30).

Текстовая интерпретация:

«*В информационной системе в приложении «Серверное приложение 1» происходит выполнение функции 1, которая по завершению подготавливает либо данные вида 1, либо данные вида 2.*

- *При выполнении условия «Если получены данные вида 1», то происходит передача этих данных через поток «Передача данных вида 1» на серверное приложение 2 для выполнения функции 2, которая возвращает на серверное приложение 1 данные обработки через поток «Результат выполнения функции 2»;*
- *При выполнении условия «Если получены данные вида 2», то происходит передача этих данных через поток «Передача данных вида 2»*

на серверное приложение 2 для выполнения функции 3, которая возвращает на серверное приложение 1 данные обработки через поток «Результат выполнения функции 3»».

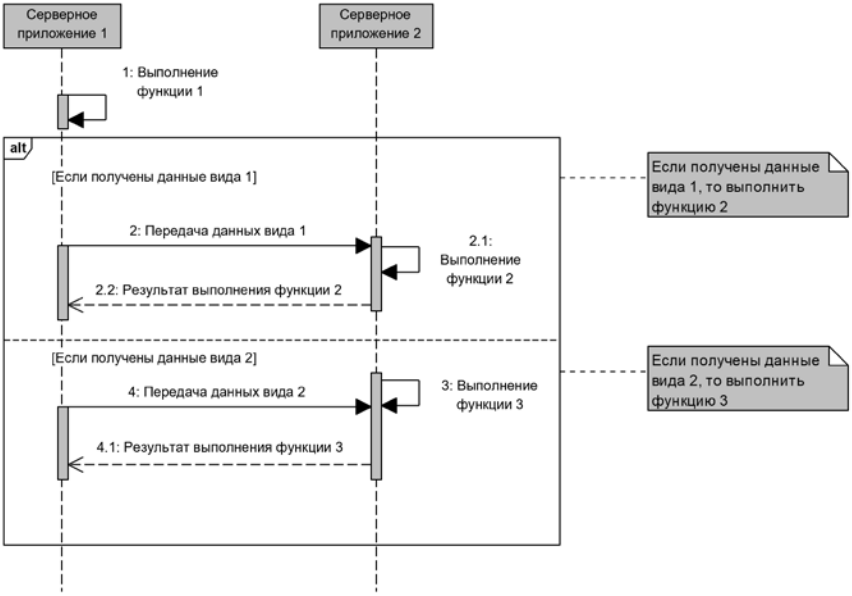


Рисунок 30 – Пример использования альтернативного фрагмента последовательностей

2. Возможный фрагмент последовательностей (англ. Optional Sequence Fragments)

Возможный фрагмент последовательностей – этот фрагмент будет исполняться только тогда, когда выполняется заданное предварительное условие. Аналогичен альтернативному фрагменту последовательностей, но имеет только один возможный вариант.

Элемент «Возможный фрагмент последовательностей» имеет обозначение «opt» (opt).

Рассмотрим использование этого фрагмента последовательностей на примере.

Пусть существует некоторая информационная система, которая включает в себя приложения: «Серверное приложение 1» и «Серверное приложение 2». При выполнении на серверном приложении 1 некоторой функции 1, возможно возникновение ошибки. Если такое событие происходит, то информация об ошибке передается на серверное приложение 2, в котором происходит обработка (коррекция) ошибочных данных, а результат возвращается обратно на серверное приложение 1. Работа серверного приложения 1 продолжается. Если ошибка при выполнении функции 1 не произошла, то элементы, входящие в возможный фрагмент последовательностей пропускаются. Пример модели, описанного процесса, показаны ниже (см. Рисунок 31).

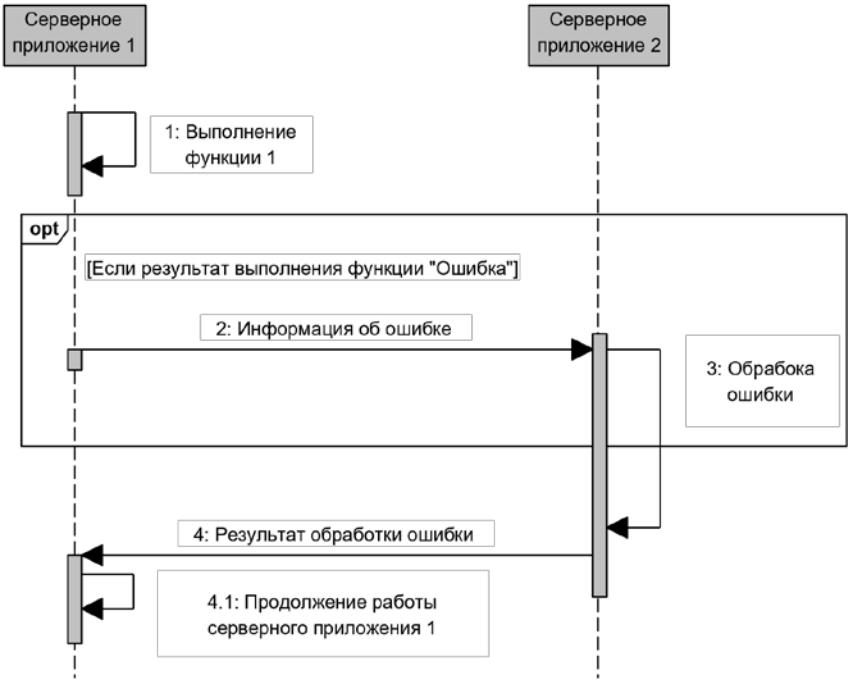


Рисунок 31 – Пример использования возможного фрагмента последовательностей

Текстовая интерпретация:

«*В информационной системе в приложении «Серверное приложение 1» происходит выполнение функции 1, которая по завершению может либо содержать ошибку, либо иметь корректные данные.*

• *Если результат выполнения функции «Ошибка», то через поток «Информация об ошибке» данные передаются в серверное приложение 2, в котором ошибка обрабатывается и через поток «Результат обработки ошибки» результат обработки передается на серверное приложение 1;*

• *Если при выполнении функции 1 ошибки не было, выполнение серверного приложения 1 продолжается без обращения к серверному приложению 2».*

3. Параллельные фрагменты последовательностей (англ. Parallel Sequence Fragments)

Параллельные фрагменты последовательностей – это фрагменты, которые исполняются параллельно.

Элемент «Параллельный фрагмент последовательностей» имеет обозначение «par» (par).

В каких случаях может понадобиться этот элемент?

На самом деле, в реальной жизни достаточно много сценариев когда нужно выполнять несколько процессов одновременно (или, другими словами, параллельно). Например, в «запарке» на работе бывает нужно одновременно говорить по телефону, печатать отчет для руководства и обдумывать план действий на завтра. Я уже молчу о женщинах, которые умудряются делать 5–6 дел одновременно.

Более серьезный пример связан с высокопроизводительными компьютерами, имеющими механизмы распараллеливания вычислений, т. е. необходимо учесть, например, что для решения некоторой задачи необходимо получить набор данных из разных процессов таким образом, чтобы эти данные были актуальными (т. е. получены в текущий момент времени) и пришли на вход вычисляющей функции практически одновременно.

Ниже рассмотрим пример, поясняющий назначение элемента «Параллельный фрагмент последовательностей» (см. Рисунок 32).

Текстовая интерпретация:

«В некотором приложении существуют четыре функции: «Функция 1», «Функция формирования данных 1», «Функция формирования данных 2», «Функция формирования данных 3».

Функции 1 в процессе работы необходимо получить данные типа: «Данные 1», «Данные 2», «Данные 3», которые должны быть сформированы одновременно (параллельно) в функциях «Функция формирования данных 1», «Функция формирования данных 2», «Функция формирования данных 3», соответственно. Функция 1 отправляет запросы на получение данных параллельно и ожидает ответа от соответствующих функций. По получению ответа, Функция 1 консолидирует (обрабатывает), полученные данные».

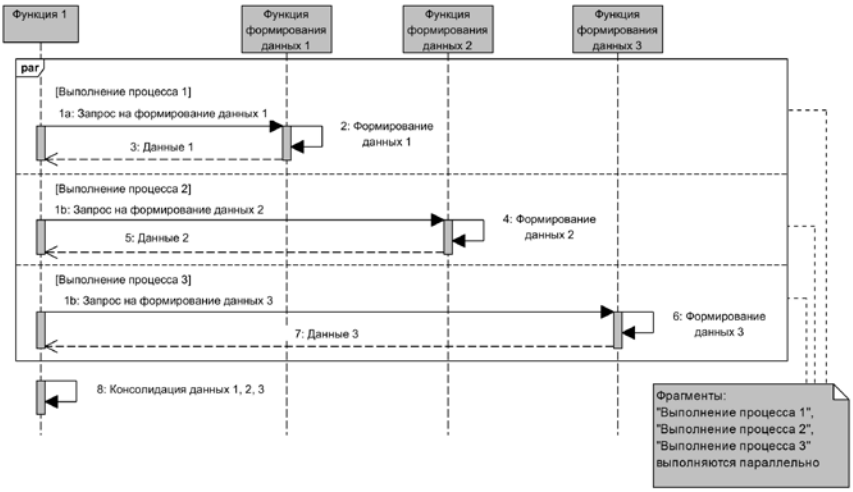


Рисунок 32 – Пример использования параллельного фрагмента последовательностей

4. Циклический фрагмент последовательностей (англ. Loop Sequence Fragments)

Возможный фрагмент последовательностей – этот фрагмент исполняется несколько раз до тех пор, пока не будет выполнено условие выхода из цикла.

Элемент «Циклический фрагмент последовательностей» имеет обозначение «loop» (loop).

Программистам хорошо известна конструкция, которая присутствует в большинстве языков программирования, называемая «цикл», т. е. выполнять программный код до тех пор, пока не будет выполнено некоторое условие, например, «индекс переменной будет больше 5» или «пока переменная А больше В» и т. п.

При моделировании ИБП циклическая деятельность также достаточно распространенный процесс. Рассмотрим простейший случай.

Продавец должен распродать скоропортящийся товар и по исчерпанию товарных запасов закрыть магазин. Предполагается, что количество покупателей всегда больше, чем количество имеющегося товара, т. е. процесс будет завершен в мыслимые временные сроки. Этот процесс можно графически отобразить так, как показано ниже (см. Рисунок 33).

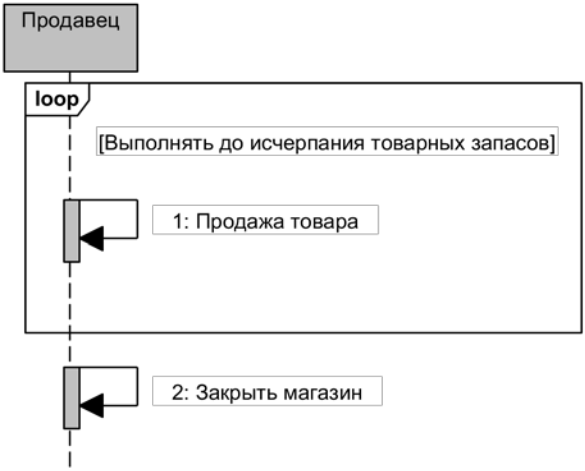


Рисунок 33 – Пример использования
циклического фрагмента последовательностей

В теле элемента «loop» необходимо указать условие (в квадратных скобках), при котором цикл завершается. В данном случае это «[Выполнять до исчерпания товарных запасов]».

Как правило, моделируемые ИБП несколько сложнее, чем показано в предыдущем примере, и, если описывать все условия, при которых должен выполняться цикл, то строка условий получится слишком длинной и плохо воспринимаемой для того, кто читает модель. А также велика вероятность появления ошибки в описании. В этих случаях используют комбинацию из нескольких фрагментов последовательностей, например, совместно с циклическим фрагментом используют альтернативный фрагмент.

Расширим предыдущий пример требованием выполнять процесс «Продажа товара» только в рабочее время, т. е. выполнять процесс «Закрыть магазин» в определенный момент времени даже в том случае, если товар не распродан. Вариант модели такого процесса показан ниже (см. Рисунок 34).

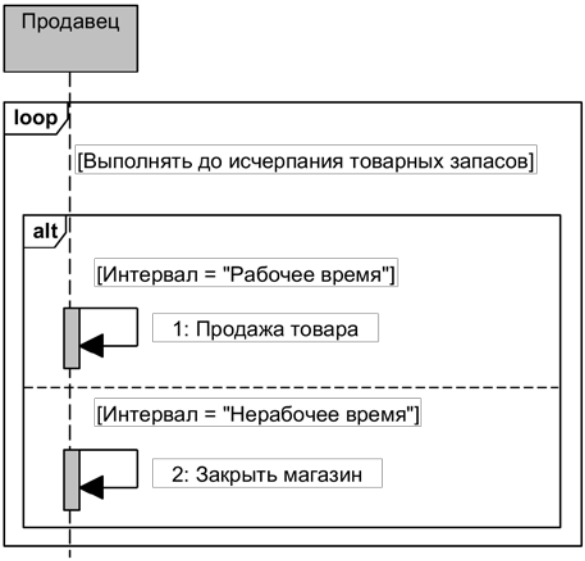


Рисунок 34 – Пример совместного использования циклического и альтернативного фрагментов

Текстовая интерпретация:

«*Сущность «Продавец» выполняет циклический процесс, который соответствует условию «[Выполнять до исчерпания товарных запасов]», и, который включает в себя два альтернативных подпроцесса:*

• *«Продажа товара» – подпроцесс выполняется, если временной интервал соответствует условию «Рабочее время»;*

• *«Закрыть магазин» – подпроцесс выполняется, если временной интервал соответствует условию «Нерабочее время»».*

5. Фрагмент последовательностей критического участка (англ. Critical Region Sequence Fragments)

Фрагмент последовательностей критического участка – этот фрагмент содержит только один поток, который выполняется только однажды.

Элемент «Фрагмент последовательностей критического участка» имеет обозначение «critical» (critical).

Наименование фрагмента звучит пугающе. Для любой системы, в общем-то, так оно и есть. Хороший бизнес-аналитик всегда должен продумать ситуацию «а, что, если…». Что, если при передаче критически важных для компании данных кто-то подключится к каналу связи? Возможен ли «бунт машин» и что делать системе в такой ситуации? …

Это конечно, скорее всего, пугалки, но пугалки, которые исключать совсем нельзя. Гораздо чаще бывает ситуация, когда происходит «падение» системы (наверняка многие из вас с этим сталкивались). Что происходит после повторного включения компьютера? Это тоже многие видели: возможно будет идти самопроверка операционной системы, возможно выскочит сообщение «Файл с ошибки подготовлен и готов к отправке…», возможно будет предложено восстановить последние несохраненные файлы, возможно … Да, много чего еще возможно. Если вы увидели подобные сообщения, то это значит, что аналитик продумывал в модели последовательностей элемент «Фрагмент последовательностей критического участка». Спасибо ему за это!

Попробуем смоделировать в виде модели SeqD то, что перечислено в предыдущем абзаце (см. Рисунок 35).

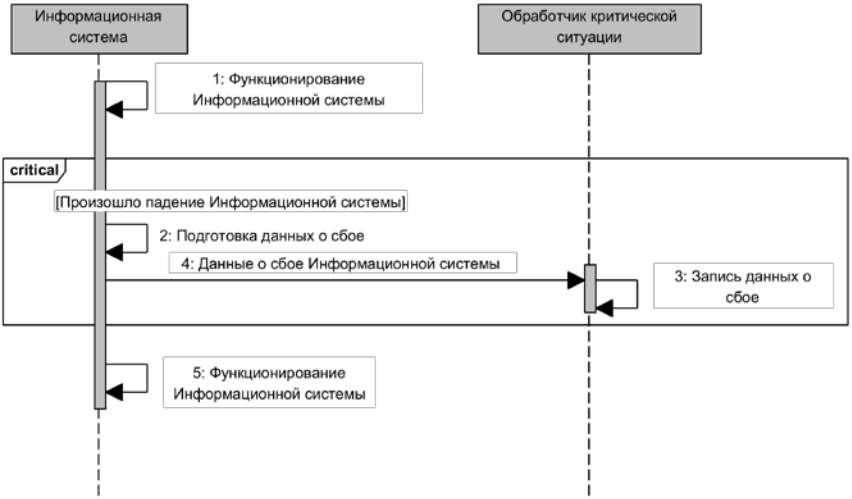


Рисунок 35 – Пример использования
фрагмента последовательностей критического участка

6. Недопустимый фрагмент последовательностей (англ. Negative Sequence Fragments)

Недопустимый фрагмент последовательностей – этот фрагмент содержит поток, который включает в себя недопустимый вывод.

Элемент «Недопустимый фрагмент последовательностей» имеет обозначение «neg» (neg).

Этот элемент подобен элементу «Фрагмент последовательностей критического участка», но, если «Фрагмент последовательностей критического участка», как правило, относится к любым сбоям в информационной системе, то «Недопустимый фрагмент последовательностей», как правило, относится к данным.

В качестве примера можно рассмотреть ситуацию, в которой некий субъект заказал доставку холодильника, а ему привезли телевизор. Последовательность «заказ холодильника – доставка телевизора» – явно недопустима, т. е. субъект должен как-то реагировать на эту ситуацию (см. Рисунок 36).

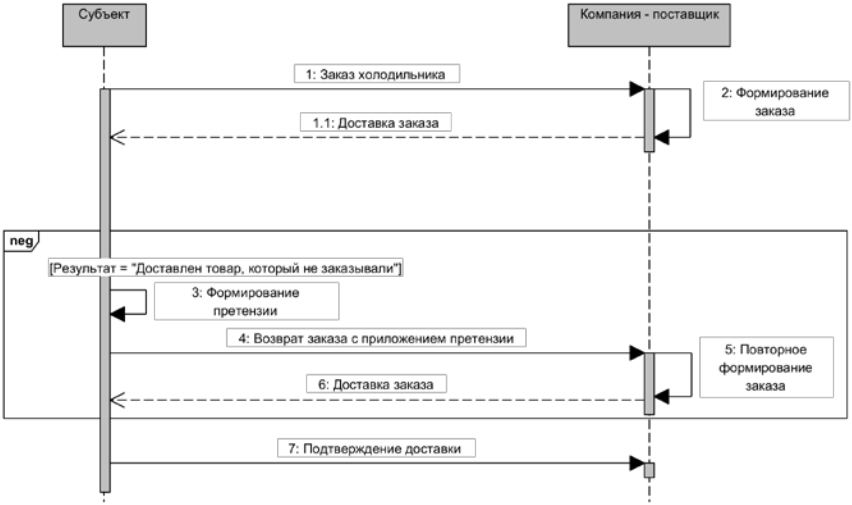


Рисунок 36 – Пример использования
недопустимого фрагмента последовательностей

Текстовая интерпретация:

«*Сущность «Субъект» через поток сообщений «Заказ холодильника» заказывает товар, который поступает к сущности «Компания-поставщик», и, которая после формирования заказа направляет обратный поток «Доставка товара». Возможна ситуация, при которой возникает результат «Доставлен товар, который не заказывали». Если такая ситуация возникла, то формируется претензия, которая направляется в компанию-поставщик совместно с возвращаемым заказом, что является основанием для повторного формирования заказа с последующей доставкой субъекту.*

Если ситуация с ошибочной доставкой заказа не возникает или ошибка обработана, то субъект направляет поток «Подтверждение доставки» в адрес компании-поставщика».

7. Ссылка на фрагмент последовательностей (англ. Reference Sequence Fragments)

До настоящего момента автор описывал элементы SeqD, которые несут функциональную нагрузку, т. е. за каждым из элементов скрывается нечто, что позволяет описать поведение

(или, говоря профессиональным языком, бихевиористическую модель) реального объекта. Однако, реальные объекты гораздо сложнее, чем те, которые приводились в примерах выше. Если мы захотим описать участников ИБП и информационные потоки, которыми они обмениваются в полном объеме (для моделей «как есть»), а также добавить новые сущности, которые будут включены в информационную систему, и информационные потоки для этих новых сущностей (для моделей «как будет»), то даже для такого простого объекта информатизации, как «информационные процессы в мастерской по ремонту обуви», получим достаточно объемную модель, которая вряд ли разместиться в читаемом виде на одном листе. Или понадобится огромный лист бумаги – даже затрудняюсь сказать какого формата…

Кроме того, модель даже одного процесса должна быть разработана для различных аудиторий читателей: для специалистов предметной области, для которых создается информационная система, и, которым не понятно, да, и не важно понимать детали технической реализации (этой аудитории важно убедиться, что вы правильно поняли, ЧТО, КАК, КТО, ГДЕ, ПОЧЕМУ, КОГДА будет выполнять, используя разрабатываемую информационную систему); для специалистов компании-разработчика, которых интересуют именно детали реализации на уровне бизнес- и системного анализа.

Специалистам ПрО (компания-заказчик) достаточно показать компоненты модели автоматизируемый процесс в виде «черных ящиков», которые имеют входы (для поступающей информации) и выходы (для преобразованной информации).

Разработчикам информационной системы требуется детализация или раскрытие инкапсуляции той структуры, которая скрыта в «черных ящиках».

Для реализации требований, изложенных выше, в нотации SeqD существует ряд элементов, которые позволяют обеспечить формирование верхнеуровневых моделей и их связь с моделями следующих уровней. Тем самым возможно обеспечение компактности отображения модели всей системы и навигация между моделями всех уровней.

Т. е. по сути мы будем говорить об элементах SeqD, которые обеспечивают связанную разметку модели.

Первым рассматриваемым элементом этого рода будет «Ссылка на фрагмент последовательностей».

Ссылка на фрагмент последовательностей – этот фрагмент является ссылкой на взаимодействие, определенное на другой диаграмме. Рамка нарисована так, чтобы покрыть линии жизни, вовлеченные во взаимодействие. Можно определить параметры и возвращаемое значение.

Элемент «Ссылка на фрагмент последовательностей» имеет обозначение «ref» (ref).

Элемент типа «ref» содержит, как уже указывалось выше, обозначение, ссылку на модель, которая инкапсулирована в элементе, возвращаемые данные и продолжительность выполнения, если она известна (см. Рисунок 37).

Рисунок 37 – Компоненты элемента
«Ссылка на фрагмент последовательностей»

Пример использования элемента типа «ref» приводится ниже.

Пусть существуют сущности, которые взаимодействуют между собой, как показано ниже (см. Рисунок 38). При внимательном взгляде можно увидеть, что первоначальное кажущееся нагромождение из элементов SeqD можно мысленно структурировать на ряд подпроцессов:

- Подпроцессы, связанные с данными типа 1–2 (см. Рисунок 39);

• Подпроцессы, связанные с данными типа 1–3 (см. Рисунок 40);

• Подпроцессы, связанные с данными типа 2–3 (см. Рисунок 41);

• Подпроцессы, связанные с данными типа 1–2-3 (см. Рисунок 42).

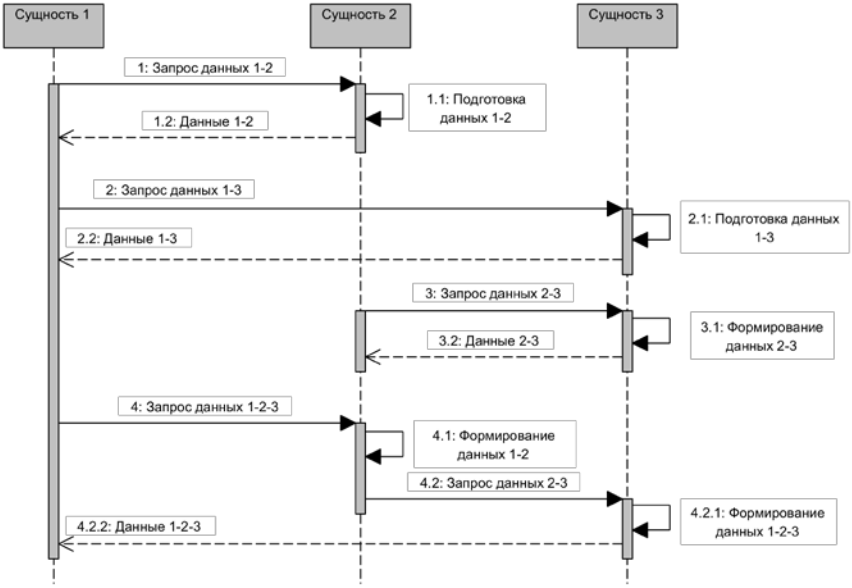


Рисунок 38 – Пример взаимодействия сущностей.
Без применения элементов «Ссылка на фрагмент последовательностей»

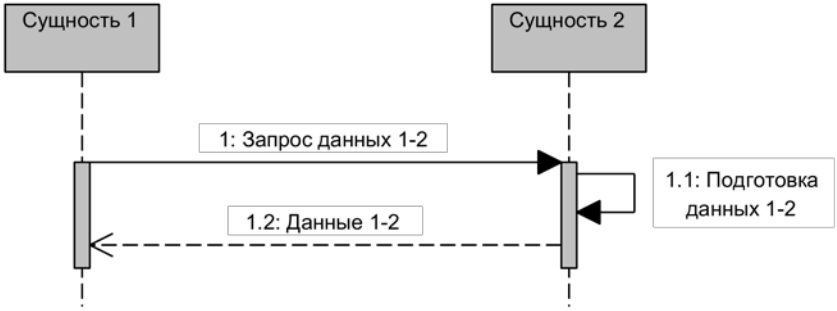


Рисунок 39 – Подпроцесс, связанный с данными типа 1–2

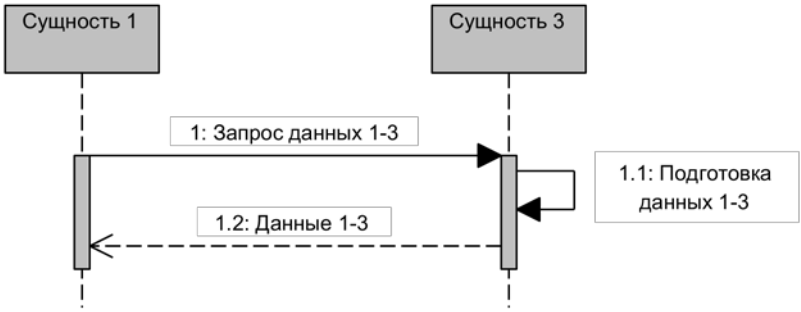


Рисунок 40 – Подпроцесс, связанный с данными типа 1–3

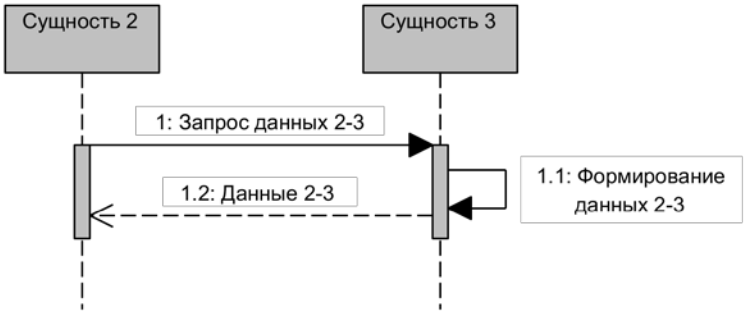


Рисунок 41 – Подпроцесс, связанный с данными типа 2–3

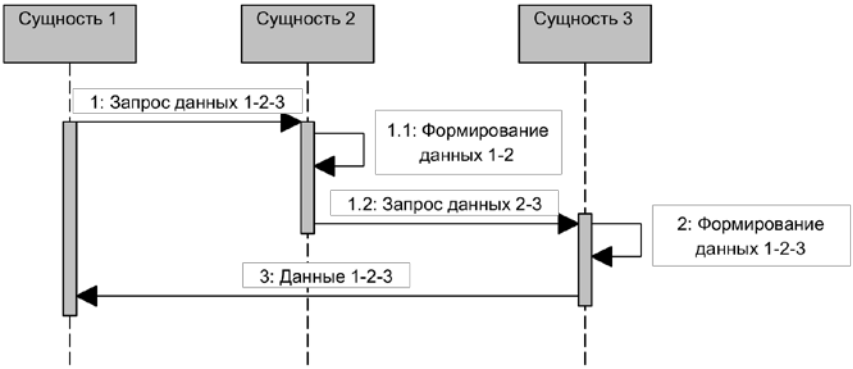


Рисунок 42 – Подпроцесс, связанный с данными типа 1–2-3

Теперь можно «упростить» модель, которую демонстрирует Рисунок 38, используя формализм элемента «Ссылка на фрагмент последовательностей» (см. Рисунок 43).

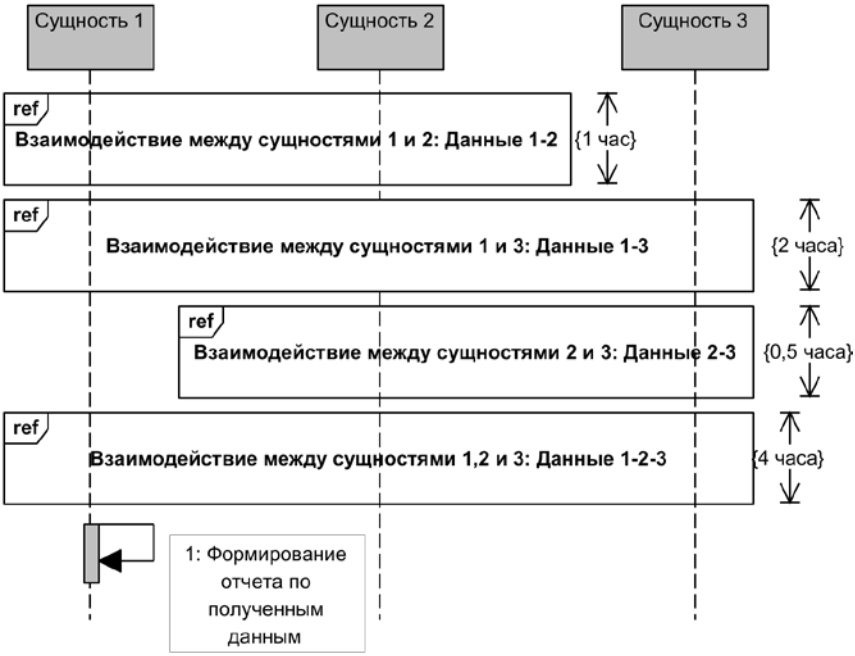


Рисунок 43 – Пример свертки модели (см. Рисунок 38) путем использования элемента «Ссылка на фрагмент последовательностей»

Читаемость модели процессов увеличилась, т. е. модель стала «прозрачнее» для читателя, который не посвящён в тонкости нотации UML. Выделено главное:

- Какие подпроцессы происходят.
- Какие сущности (линии жизни) участвуют в каждом из подпроцессов.
- Какие данные будут получены в результате.
- Продолжительность каждого из подпроцессов.

Обратите внимание: подпроцесс «Взаимодействие между сущностями 1 и 3» используют только «Сущность 1» и

«Сущность 3». На модели этот факт отражается следующим образом: линия жизни для «Сущности 2», которая не участвует в подпроцессе оказывается закрытой соответствующим элементом «ref».

8. Диаграмма последовательностей (англ. Sequence Diagram)

Есть еще один элемент в SeqD, который называется также, как и сам тип моделей – «Диаграмма последовательностей». Оставлю такое название элемента на совести авторов нотации UML. Хотя… Можно было бы и тщательнее отнестись к выбору наименований элементов, т. к. это один из важных элементов, который позволяет развить философию инкапсуляции в моделях SeqD. Будем следовать структуре книги и введем определение элемента «Диаграмма последовательностей» (повторюсь – не путать с одноименным типом моделей).

Диаграмма последовательностей – этот элемент используется для ограничения диаграммы последовательностей конкретного процесса при моделировании взаимодействия между процессами [9]

Элемент «Диаграмма последовательностей» имеет обозначение «sd» и включает в заголовок наименование процесса, который в элементе инкапсулирован (например, sd ИБП Руководителя). Синонимом, который часто используют в конкретных графических инструментах для моделирования (далее – Моделлер) в нотации SeqD, для элемента «Диаграмма последовательностей» является «фрейм». Далее по тексту автор будет использовать именно «фрейм».

Хочу заметить, что в «правильных» Моделлерах, даже в том случае, если в проекте используется одна единственная SeqD, модель будет ограничена фреймом с заголовком «sd» и соответствующими элементами внутри фрейма.

[9] Определение, которое дано в документе [3], но дополнено автором, т. к. в оригинале дается слишком пространное пояснение на множестве страниц и в нескольких разделах.

Почему же фрейм выделен, как отдельный элемент в нотации?

Все дело в том, что, как правило, этот элемент используется в моделях, в которых главным является демонстрация разнородных процессов и взаимодействия между ними. Такие модели еще называют «Диаграммы взаимодействия».

Рассмотрим вырожденный случай, которому соответствует один единственный процесс, как показано ниже (см. Рисунок 44).

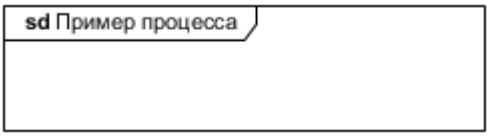


Рисунок 44 – Пример одиночного фрейма

А если процессов несколько? В таком случае используется совместно с фреймом дополнительный элемент, так называемый «шлюз», который графически представляет собой квадрат с поясняющей надписью. Шлюз – это ссылочный элемент, который присутствует в модели нижнего уровня и соединяется с соответствующим элементом модели. В границах фрейма верхнего уровня шлюзы соединяются между собой информационными потоками, которые соответствуют потокам между различными типами процессов. Т. е. можно показать группы информационных процессов и то, какой информацией они обмениваются, не вдаваясь в подробности. Для тех, кому подробности нужны, используется ссылочный механизм на модели нижнего уровня (конечно, если это оправдано логикой, создаваемого документа).

Рассмотрим следующий пример.

В некоторой организации существует два типа сотрудников: «Руководитель» и «Исполнитель». У каждого из них существуют свои ИБП: «ИБП Руководителя» и «ИБП Исполнителя». Между ИБП существуют информационные связи: «Запрос отчета» и «Отчет». Взаимодействие осуществляется через шлюзы:

• Для «ИБП Руководителя» – передающий шлюз «Передача данных от руководителя» и принимающий шлюз «Получение отчета руководителем»;

• Для «ИБП Исполнителя» – передающий шлюз «Передача отчета руководителю» и принимающий шлюз «Прием данных от руководителя».

Возможная диаграмма взаимодействия показана ниже (см. Рисунок 45).

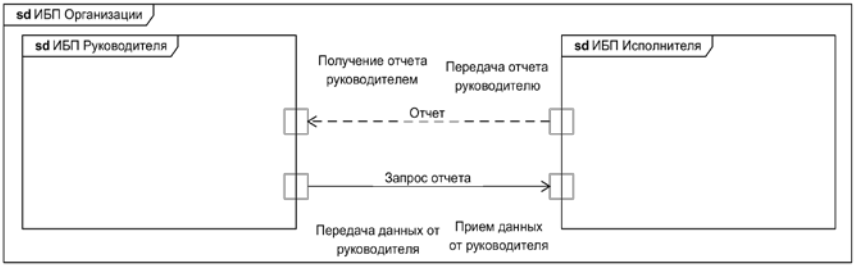


Рисунок 45 – Пример диаграммы взаимодействия между фреймами

Примеры раскрытия фреймов и использование шлюзов в моделях нижнего уровня демонстрируют Рисунок 46 и Рисунок 47 для фреймов «ИБП Руководителя» и «ИБП Исполнителя», соответственно.

Рисунок 46 – Пример раскрытия фрейма «ИБП Руководителя»

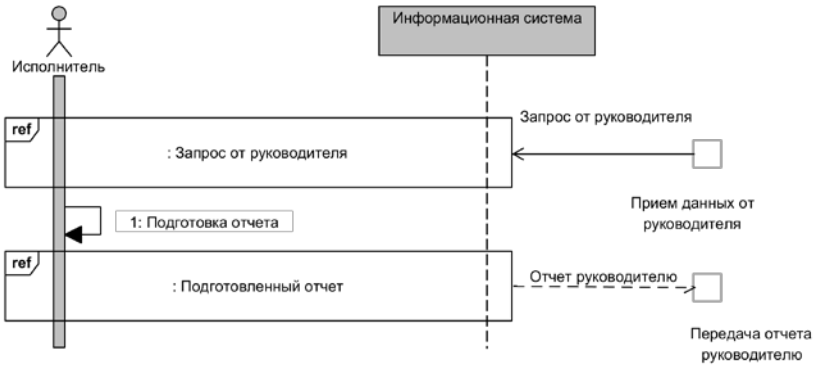


Рисунок 47 – Пример раскрытия фрейма «ИБП Исполнителя»

ix. Элемент «Продолжение» (англ. Continuation)

Рассмотрим еще один элемент, входящий в нотацию SeqD, который позволяет обеспечить читаемость модели по вертикали. Можно представить ситуацию, при которой не удается сгруппировать информационные потоки в соответствии с некоторыми признаками (например, по функциональности, по принадлежности к некоторой линии жизни и т. д.) или такая группировка только усложнит и запутает модель. Т. е. нужно упорно отображать потоки между линиями жизни, опускаясь вдоль этих линий все ниже и ниже. Приходим к ситуации, рассмотренный выше – размещению модели, которая должна вписаться в мыслимые форматы страниц документа. Или получится длиннющий свиток, который, разве что, понравился бы в эпоху Средневековья – в те времена документ в виде свитка был обычным делом.

Рассмотрим элемент «Продолжение», который решает эту проблему, а, по сути аналогичен тому, как поступают в редакции журнала, если статья, рассказ, роман и т. п. слишком большие для размещения в одном номере издания – на самом интересном месте пишут «Продолжение следует» или «Продолжение в следующем номере».

Элемент «Продолжение» – это семантический способ для определения продолжения модели на другой диаграмме для заданных линий жизни

Отображение элемента «Продолжение» показывает Рисунок 48.

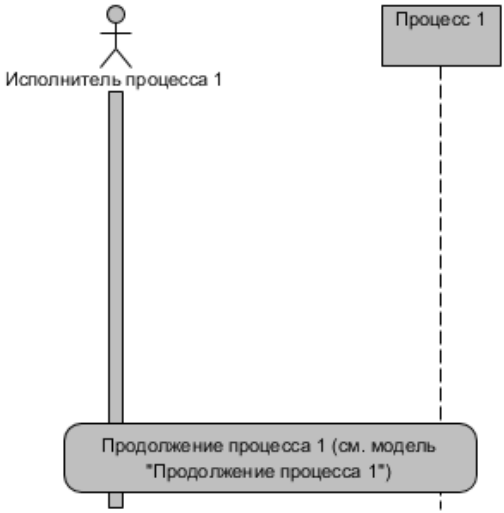


Рисунок 48 – Пример элемента «Продолжение»

Элемент «Продолжение», как видно из рисунка, покрывает линии жизни, для которых описание процессов будет продолжено на другой диаграмме. Нотация для SeqD допускает (теоретически) любой текст внутри элемента, но рекомендуется писать, аналогично журнальной практике, текст, который прямо указывает, что это еще не все и будет продолжение на другой диаграмме с указанием наименования этой диаграммы.

В качестве примера использования элемента «Продолжение», приведем группу моделей для следующего ИБП:

Существует два независимых подпроцесса «Процесс 1» и «Процесс 2», которые выполняют два участника «Исполнитель процесса 1» и «Исполнитель процесса 2», соответственно. В каждом из подпроцессов выполняются стандартные обмены информацией: исполнитель отправляет данные для процесса, в процессе формируются результаты, которые

отправляются обратно к исполнителю, исполнитель отправляет полученные данные для проверки, выполняется проверка данных, результаты проверки возвращаются исполнителю.

Отображение модели этого ИБП с использованием элемента «Продолжение» приводится ниже (см. Рисунок 49).

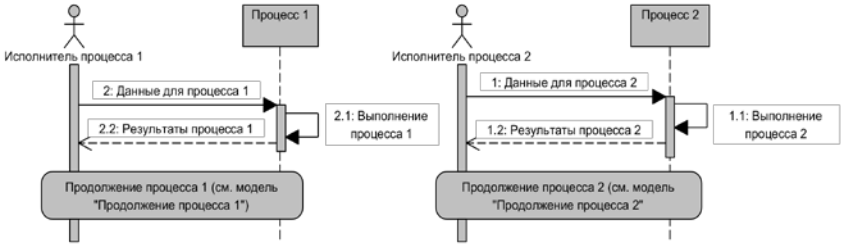


Рисунок 49 – Вариант модели для процессов, описанных выше, с использованием элемента Продолжение»

Переход по ссылкам, указанным в элементе «Продолжение», приведет к моделям, показанным ниже (см. Рисунок 50 для «Процесс 1» и Рисунок 51 для «Процесс 2», соответственно).

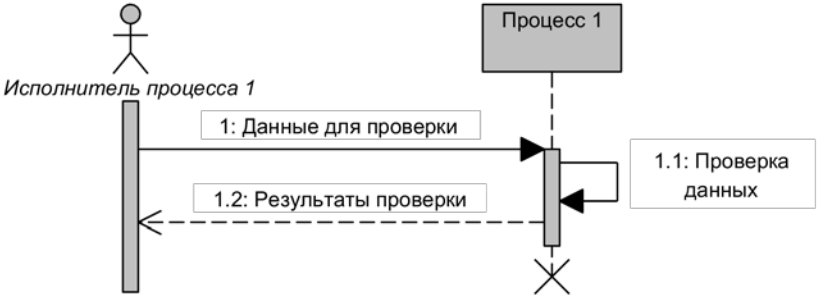


Рисунок 50 – Пример модели «Продолжение процесса 1» для ссылки «Продолжение процесса 1 (см. модель «Продолжение процесса 1»)»

Рисунок 51 – Пример модели «Продолжение процесса 2» для ссылки «Продолжение процесса 2 (см. модель «Продолжение процесса 2»)»

D. Выводы к ГЛАВЕ 1

В настоящем разделе читатель познакомился еще с одной добродетелью, которой должен обладать бизнес-аналитик – знание нотации UML в объеме, который позволит передать плоды своего труда остальным членам группы разработки, в частности, ближайшему коллеге в технологической цепочке, а именно: системному аналитику. Возможно, что читатель работает на небольшом проекте с небольшим бюджетом и размеры группы ограничены[10]. Т. е. было решено, что роли бизнес-аналитика и системного аналитика объединяются. В таком случае, от читателя потребуется знание UML в полном объеме, вы-

[10] Хочу заметить, но это личное мнение автора, что не стоит экономить и объединять роли бизнес-аналитика и системного аналитика, замыкая это на одном человеке, т. к. для непосвященного (а это, к сожалению, в большинстве случаев, человек, который принимает решения) и бизнес-аналитик, и системный аналитик – это одно и тоже. Со стороны похоже – и тот, и другой рисуют какие-то картинки, которые никому (т. е. ему – принимающему решение) непонятны, а, значит с этим справится и один человек – все равно программисты чего-нибудь придумают и без этих умников.

Хотелось бы напомнить лицам, принимающим решение, что, как минимум, бизнес-аналитик своей основной задачей имеет понимание нужд функционального заказчика, которое он должен донести до остальных членов группы разработки в виде однозначно трактуемых моделей. А системный аналитик должен это понимание расширить до формализма, который ОБЯЗАНЫ понимать программисты, т. е. UML-модели.

Подозреваю, что это достаточно благостное видение современной ситуации с моей стороны – эту книгу не будут читать, скорее всего, ни лица, принимающие решение, ни программисты….

ходящем за рамки того, что описано в настоящей книге. Но это совсем другая история…

Останемся в границах концепции настоящей книги: читатель является бизнес-аналитиком (или хочет таковым стать) и другую роль не выполняет.

1. Бизнес-аналитик после формирования моделей в нотации BPMN формирует в нотации UML модель вариантов использования, в которой явно указывается:

   a. Классификация и состав участников, участвующих в ИБП до автоматизации (модель «как есть»), и, участвующих в ИБП после автоматизации (модель «как будет»);

   b. Устанавливаются границы системы;

   c. Четко прописываются варианты использования системы каждым из участников, а также то, что каждый из элементов из вариантов использования содержит при обращении к нему участника (связи «include» или, что допустимо, русский вариант «включение») (т. е. другие варианты использования, к которым участник напрямую не обращается) или может к себе подключать при необходимости (связи «Extend» или, что допустимо, русский вариант «Расширение»);

2. Поскольку модель «варианты использования» показывает все мыслимые варианты сценариев, которые может выполнять каждый из участников, но не показывает порядок их выполнения, требуется показать ЧТО именно и КОГДА именно, включая допустимую продолжительность процесса, требуется явно указать порядок взаимодействия каждого из участников друг с другом и с проектируемой системой[11]. То есть бизнес-аналитик после формирования моделей в нотациях BPMN и UML (модель типа «варианты использования») должен сформировать в нотации UML модель последовательностей, в которой явно указывается:

   a. КТО или ЧТО является инициатором процессов.

   b. Какими сообщениями обмениваются элементы системы.

---

[11] Как сказали бы специалисты в области аппаратных средств: «расписать работу системы по тактам».

c. При каких условиях может быть отправлено сообщение (как вырожденный случай – отсутствие условий).

d. Когда обмен сообщениями может быть циклическим.

e. Как обрабатываются недопустимые сообщения или фатальные сбои системы.

E. Вопросы и задачи для повторения материала

1. Используя элемент «Обобщение», попробуйте сформировать модель пользователя информационной системы для небольшого магазина.

2. В нотации моделей «Варианты использования» сформируйте сценарии для каждого из пользователей из задачи 1. Учтите возможность того, что некоторые варианты использования могут быть общими для различных пользователей. Опишите созданную модель на естественном языке.

3. Используя результаты решения задач 1 и 2, постройте модель «Последовательности». Необходимо использовать элементы группы «Фрагменты последовательностей». Опишите созданную модель на естественном языке.

ГЛАВА 2.
ВЗГЛЯНУТЬ ГЛАЗАМИ ПОЛЬЗОВАТЕЛЯ. ПРОТОТИПИРОВАНИЕ ИНТЕРФЕЙСОВ

- **Определение экранных и печатных форм**
- **Выбор технологии реализации**
- **Компоновка экранных форм**
- **Презентация концепта информационной системы**
- **Выводы**

Можно идеально строить модели ИБП, используя всю мощь нотаций, которые рассматривались в этой книге: BPMN. ERD, UML, но конечный пользователь, для которого и затевается разработка, не увидит того, что же должно быть в результате. Точнее, конечный пользователь увидит те ИБП, которые он выполняет, и, которые он будет выполнять после автоматизации его деятельности, увидит данные, которые он будет получать, и, которые должен формировать. Конечный пользователь также увидит информационные потоки, которые будут направлены к нему, и, которые он будет порождать сам. Но, из этого не будет видно, как, сидя за своим рабочим местом, пользователь получит доступ ко всем, красиво разрисованным, ресурсам.

Еще одна тонкость. Теоретически можно отобразить через модели все тонкости работы с информационной системой, но сколько это займет времени… Конечно, не в такой степени, но это будет очень напоминать то, что однажды сказал Р. Фейнман: «Все на свете можно описать системой дифференциальных уравнений, но вопрос – кто сумеет решить эту систему и сколько на это понадобится времени…». Нужно понимать в какой момент остановиться с моделированием и перейти к практике, т. е. наглядно показать функциональному заказчику, КАК будет концептуально выглядеть система: какие экранные и печатные формы будут доступны конечному пользователю, какие кнопки и клавиши будут в экранных формах, в каком порядке можно эти кнопки и клавиши нажимать, какая автоматизация действий пользователя будет предусмотрена при

вводе информации, как будет осуществляться навигация между формами, как будет осуществлена «защита от дурака» и многое другое.

Какого бы высокого мнения о своих способностях не был бизнес-аналитик, но нужно иметь ввиду: он не эксперт в ПрО, процессы которой автоматизирует (хотя бывают исключения), а следовательно – в том, что демонстрируется в качестве концепции заказчику, будут и ошибки, и пробелы, и недоработки. Бизнес-аналитик – это не бухгалтер, не врач, не директор, не …. Бизнес-аналитику не нужно каждый день садиться за компьютер, чтобы просматривать и формировать, например, бухгалтерские документы, электронные карты пациентов, отчеты и т. п. Отсюда вывод: прислушайтесь к мнению конечного пользователя, обсудите с ним детали и внесите правки в ваше «творчество» – не вам потом каждый день видеть убогую (с точки зрения конечного пользователя) экранную форму. И представьте еще то, КАК будет в адрес разработчиков высказываться конечный пользователь. Самое мягкое будет: «Какой придурок это разрабатывал…».

Очевидный способ достижения согласия в видении будущей информационной системы между функциональным заказчиком и разработчиками – представить для обсуждения концептуальное представление будущей информационной системы. Не так давно для этого приходилось выполнять разработку программного обеспечения с минимальными функциональными возможностями, главным образом, для демонстрации пользовательского интерфейса и его поведения при взаимодействии с оператором. На это уходило время и не факт, что полученное решение нравилось заказчику. Вносились замечания и, как следствие, необходимо было вносить правки в код (в лучшем случае) или полностью переделывать концепт будущей информационной системы. Цикл мог иметь достаточно много повторений.

В настоящее время есть достаточно много способов достижения согласия между заказчиком и разработчиком. Одним из таких способов является прототипирование интерфейсов, суть которого сводится к следующему.

А. Определение экранных и печатных форм

В соответствии с результатами бизнес-анализа определяются экранные формы и соответствующие процессы или группы процессов (на основе моделей BPMN), а также варианты использования (на основе моделей типа «Варианты использования» нотации UML), которые будут соответствовать этим формам. Также определяются печатные формы, которые будут привязаны к соответствующим экранным формам. Если предполагается предпросмотр печатной формы, то определяется экранная форма, которая будет визуализировать печатный документ до вывода на печать.

В соответствии с моделями типа «Последовательности» нотации UML и соответствующими моделями BPMN формируется граф, узлами которого являются экранные и печатные формы, а дугами является последовательность, в соответствии с которой эти формы будут взаимодействовать (пример см. Рисунок 52).

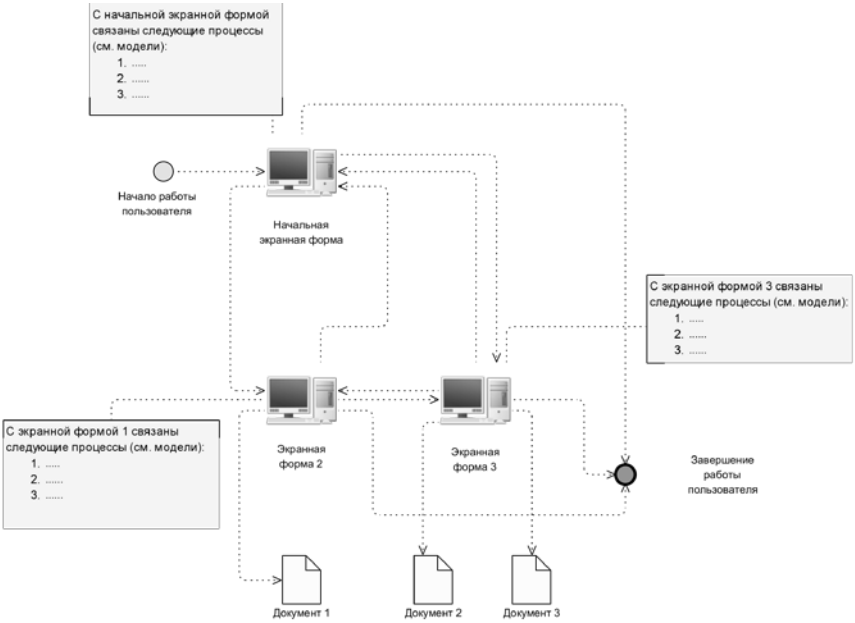


Рисунок 52 – Пример графа взаимодействия экранных форм информационной системы

Представленный пример является одним из вариантов отображения графа взаимодействия. Возможно и другое представление, которое может зависеть от вкусов и предпочтений бизнес-аналитика, а также от возможностей графического редактора, который используется.

B. Выбор технологии реализации

Лет 40 назад такой вопрос – «Выбор технологии реализации» – просто бы, скорее всего, не стоял. Существовал один вариант: однопользовательская операционная система, на которой устанавливался толстый клиент, взаимодействующий с данными локально или на сервере.

Такой вариант существует и в настоящее время, но разработчики (в соответствии с пожеланиями заказчика) в основном тяготеют различным web-технологиям: облачные вычисления, клиент-серверные приложения, в которых пользовательский интерфейс обеспечивает тонкий клиент и т. п.

От того, какая технология будет выбрана, будет, в значительной степени, зависеть вид интерфейса пользователя. И этот момент нужно держать в уме, т. к., возможно, ваш графический редактор не сделает разницы в том, как выглядит тот или иной элемент в зависимости от выбранной технологии. Для концепта формы это и не важно, но, как будет выглядеть окно (в толстом или тонком клиентах), как правило, большинство редакторов различают. Разработчики редакторов пользовательского интерфейса также, как правило, делают различие для различных операционных систем и вида устройств: стационарный компьютер, ноутбук, планшетник, мобильное устройство.

Эта особенность демонстрируется ниже (см. Рисунок 53 … Рисунок 58).

Автор для примеров использовал возможности Системы разработки ИТ для бизнес-систем Visual Paradigm (см. https://www.visual-paradigm.com/).

Рисунок 53 – Пример фрейма экранной формы для смартфона на платформе Android

Рисунок 54 – Пример фрейма экранной формы для планшетника на платформе Android

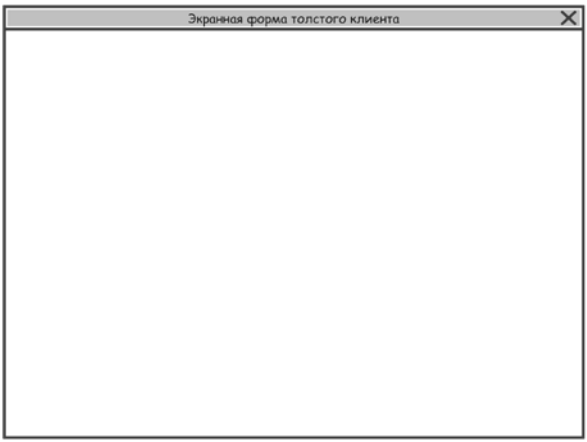


Рисунок 55 – Пример фрейма экранной формы толстого клиента для стационарного компьютера

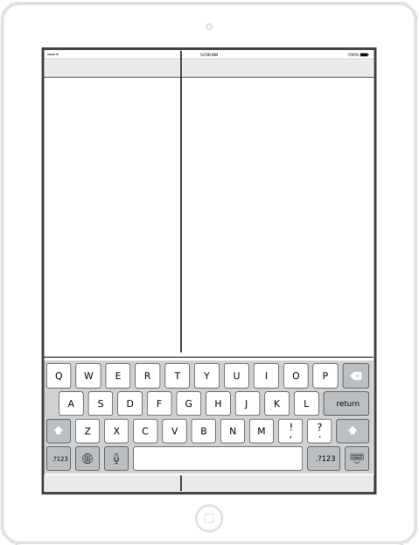


Рисунок 56 – Пример фрейма экранной формы для планшетника на платформе iPad

Рисунок 57 – Пример фрейма экранной формы для смартфона на платформе iPhone

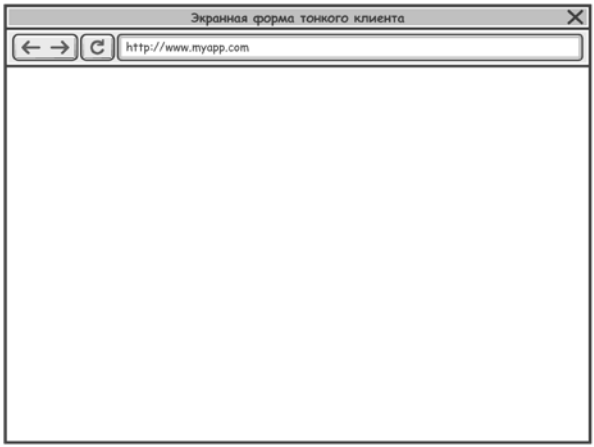


Рисунок 58 – Пример фрейма экранной формы тонкого клиента для стационарного компьютера

C. Компоновка экранных форм

При компоновке экранных форм необходимо учитывать ряд правил, которые помогут разработать правильный концепт формы:

1. Форма должна содержать все элементы, которые обеспечат функциональность, определенную в соответствующих аналитических моделях, например, формирование нового документа, просмотр, редактирование и удаление существующего документа, получение уведомлений и т. д.;

2. Форма должна содержать элементы, которые явно или не явно обеспечивают навигацию между другими формами, определенными согласно рекомендаций, изложенных в разделе ГЛАВА 2.а. Например, можно расположить в форме клавишу «Переход к следующей форме» (явная навигация), можно определить функциональность для клавиши «Новый документ», в соответствии с которой будет вызвана форма, определяющая шаблон для нового документа (не явная навигация);

3. Каждый элемент, вводимый в форму должен иметь краткий комментарий, который поясняет разработчику или пользователю для чего этот элемент введен в концепт;

4. При необходимости, должен быть обязательно включен элемент, позволяющий вернуться к предыдущей форме или к начальной (стартовой) форме;

5. На каждой форме обязательно должен содержаться элемент, вызывающий контекстную справку для данной формы;

6. Необходимо продумать возможность блокировки элементов, которые могут быть активными только при выполнении определенных условий. Например, если не выбран в списке существующий документ, то элемент (чаще всего клавиша) «Удалить» должен быть заблокирован[12];

7. Необходимо помнить, что пользователь, в силу физиологии зрения, видит не больше 40% формы по центру, т. е.

---

[12] В 80-е годы такую технологию называли «Технология молотка», поскольку для человека, который даже впервые в жизни увидел молоток, интуитивно ясно для чего нужен молоток и, как взять его в руки.

элементы, расположенные на периферии формы, пользователь, как говорится, «в упор не видит»;

8. Не следует перенасыщать форму, т. к. существует ряд исследований, которые показывают – человек, рассматривая изображение (а форма – это специфическое изображение), воспринимает не более шести значимых элементов;

9. Не пытайтесь заниматься дизайном: выбор цветовой гаммы, шрифт элементов и их размеры и т. п. – это не работа бизнес-аналитика. Этим, после согласования концепта с заказчиком разработки, должен заниматься дизайнер[13]. Просто запомните дизайнерскую мудрость: «Черный и белый цвета – друзья дизайнера».

Пример формы, скомпонованной в соответствии с правилами, изложенными выше, а также ее реакция на выбор пользователя показан ниже (см. Рисунок 59 … Рисунок 61).

[13] Если вы разрабатываете информационную систему для документооборота в авиастроительной компании – не приглашайте дизайнера, который раньше разрабатывал сайты для кошатников… Результат может быть самым аховым.

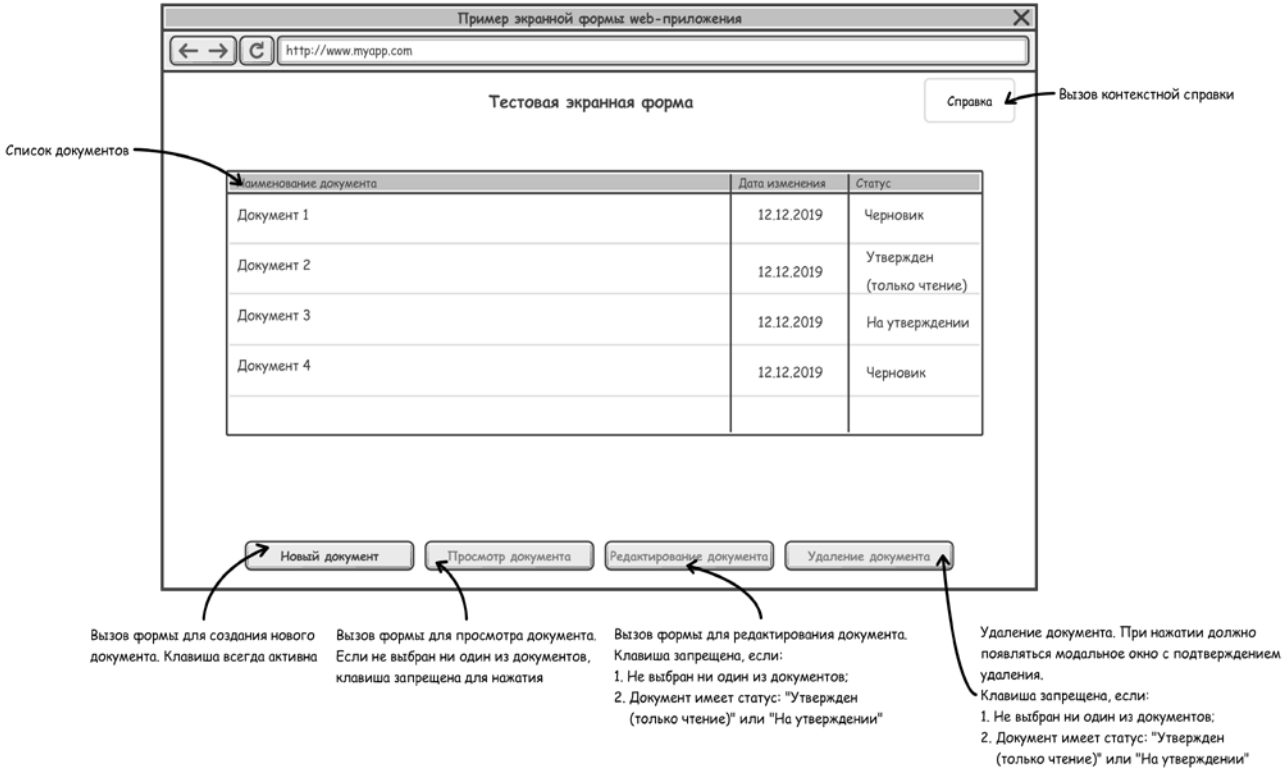


Рисунок 59 – Пример компоновки концепта экранной формы

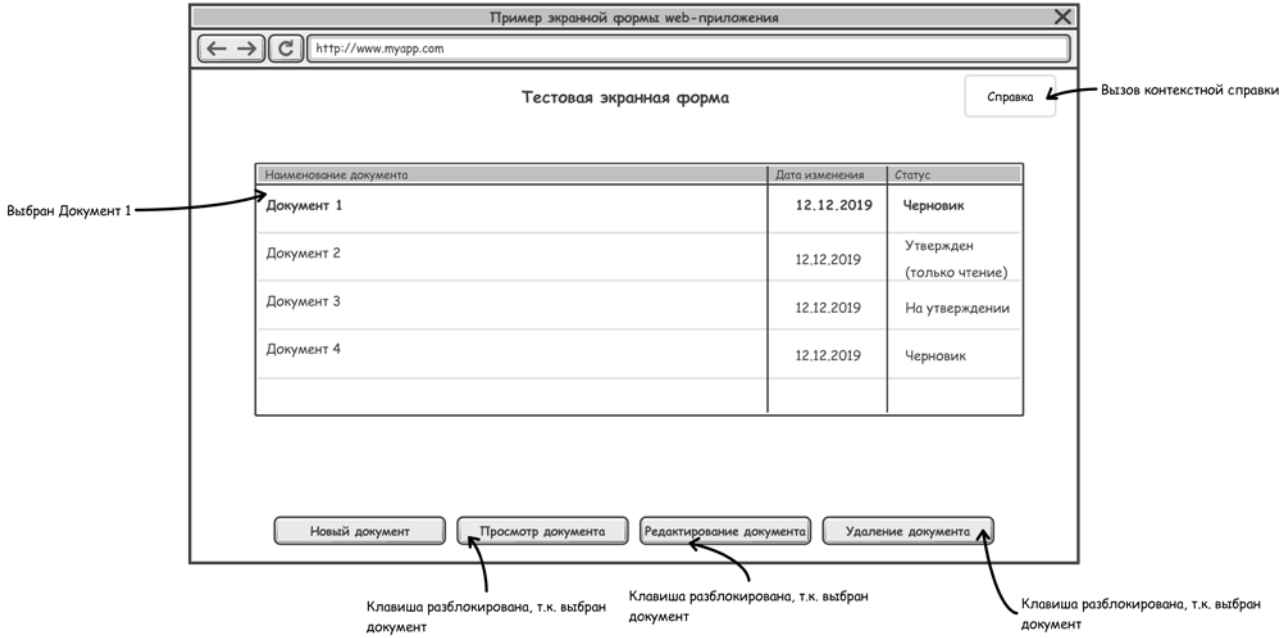


Рисунок 60 – Пример поведения концепта экранной формы при выборе документа со статусом «Черновик»

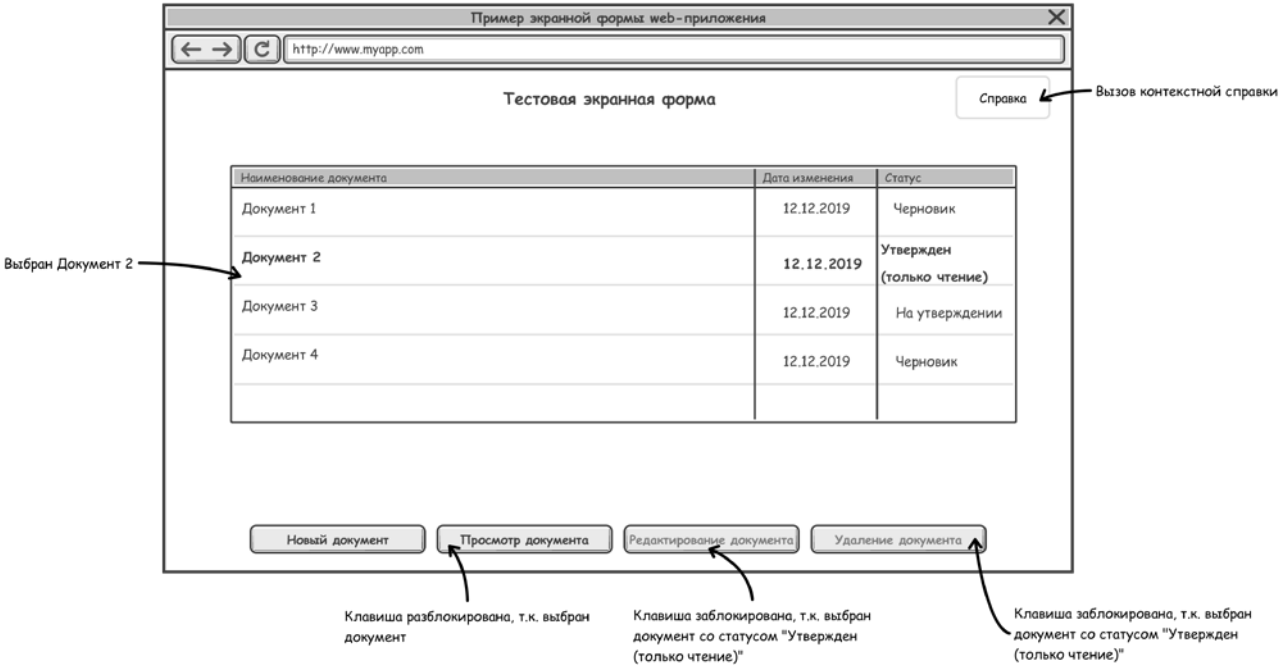


Рисунок 61 – Пример поведения концепта экранной формы при выборе документа со статусом «Утвержден (только чтение)»

D. Презентация концепта информационной системы

После подготовки и согласования моделей ИБП, включая модели данных, после формирования концептов экранных форм, самое время доложить заказчику о своем видении того, что будет. И доложить так, чтобы заказчик не смотрел на модели и схемы со «стеклянными» глазами, а понимал действительно, ЧТО ему предлагают, а специалисты заказчика могли высказать свое мнение и внести, при необходимости, правки и предложения.

Идеальным вариантом была бы презентация, на которой была бы показана работающая информационная система, но в которую можно было бы мгновенно вносить правки. Но это вряд ли возможно.

Обычная практика состоит в том, чтобы получить набор графических образов экранных форм, подобных рассмотренным в предыдущем разделе, включить их в файл презентации в соответствии с некоторым сценарием показа и продемонстрировать все это перед заказчиком. Такой подход имеет ряд недостатков:

1. Как правило, возможных сценариев сложной информационной системы для различных групп пользователей может быть достаточно много. Предусмотреть все возможные сценарии в обычной презентации долгая и неблагодарная задача. Что-нибудь не будет учтено, а кто-нибудь из будущих пользователей попросит показать именно этот сценарий. Придется выкручиваться по ходу презентации, прыгая со слайда на слайд.

2. Если в процессе презентации возникнут правки по виду экранных и печатных форм, то придется записать эти правки, затем, вернувшись в офис, правки учесть, внести исправленный вариант в презентацию (скорее всего, в несколько слайдов) и повторить презентацию с участием представителей заказчика. И никто не даст гарантии, что не возникнут новые правки или то, как вы исправили формы в соответствии с предыдущими замечаниями. А это повторение согласования… Процесс может затянуться.

Второй путь презентации состоит в том, чтобы концепты экранных и печатных форм были подготовлены в специализированном графическом редакторе, который позволяет не только скомпоновать экранную или печатную форму, но и имитировать

их работу в составе информационной системы. Такой подход имеет ряд преимуществ:

1. Сам процесс разработки презентации позволяет бизнес-аналитику увидеть недостатки и просчеты в концепции, например, нехватка элементов навигации (или наоборот – лишние), тупиковые ветви сценариев выполнения ИБП или, наоборот, узлы графа сценария с неопределенностью условий перехода к другому узлу (точнее – форме), большая плотность элементов формы (т. е. задуматься над вопросом о возможном разбиении одной формы на несколько), возникновение возможных исключений, которые никак не обрабатываются системой, и т. д., и т. п.

2. Отсутствует необходимость одну и ту же картинку размещать на нескольких слайдах для демонстрации различных сценариев – в имитационной модели можно просто вернуться к нужной форме, также как в реальной информационной системе – нажимая клавиши навигации;

3. В процессе презентации, если возникают замечания и предложения со стороны заказчика, можно остановиться и внести правки в экранные и печатные формы, т. е. процесс согласования сокращается до минимума.

Существуют ли такие «волшебные» инструменты, которые позволяют готовить такие презентации? Автору не хотелось бы делать рекламу кому бы то ни было, но, похоже, не обойтись.

В составе инструментов проектирования, которые предоставляет Система разработки ИТ для бизнес-систем Visual Paradigm (см. https://www.visual-paradigm.com/), существует особый тип диаграмм, так называемые «Диаграммы компоновки» (англ. Wireframe Diagram). Ниже показана диаграмма компоновки, построенная на основе экранных форм из ГЛАВА 2.с (см. Рисунок 62).

На диаграмме видны экранные формы и связи между ними. Кроме того, присутствует элемент «Decision», задачей которого является обработка действий, выполняемых в форме «Начальная форма».

К великому сожалению, формат книги не позволяет запустить на выполнение, показанную диаграмму, но поверьте на слово автору: все «пряники», описанные выше, присутствуют.

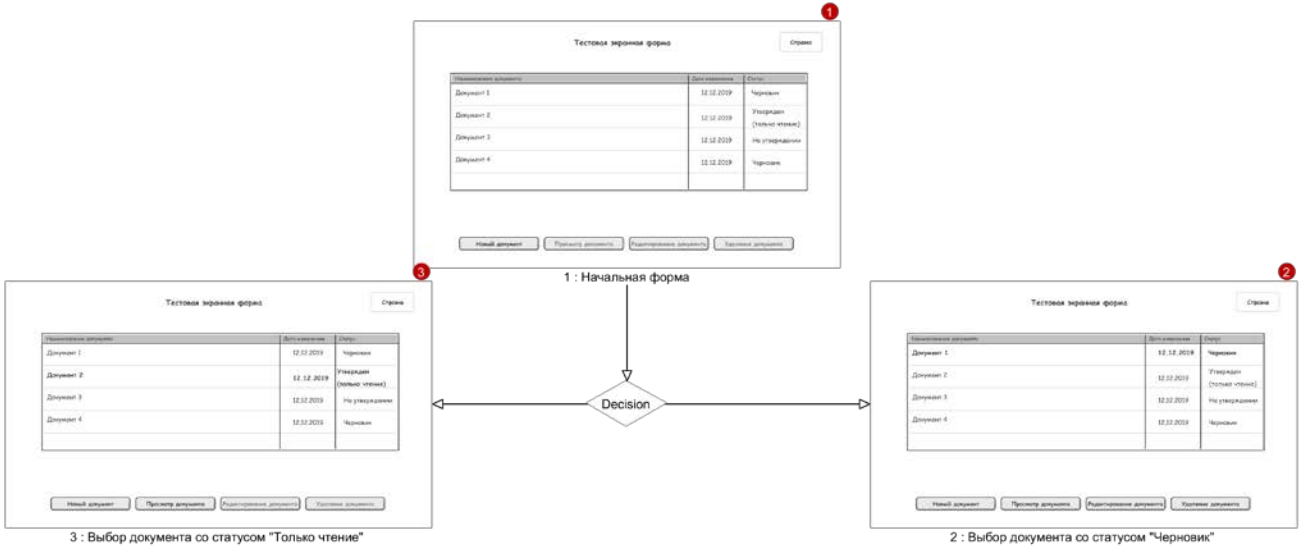


Рисунок 62 – Пример диаграммы компоновки

E. Выводы к ГЛАВЕ 2

В настоящем разделе читатель получил представление о «финишных» операциях, которые выполняет бизнес-аналитик на этапе проектирования информационной системы: формирование концептов экранных и печатных форм, а также их компоновка при подготовке презентации.

К сожалению, формат книги не позволяет «оживить» этот интересный и творческий процесс. Это, с одной стороны. С другой стороны, графических редакторов великое множество и каждый выбирает тот, который ему нравится (к огорчению автора, на практике, выбирают не действительно хороший инструмент, а тот, который бесплатный – в результате имеем то, что имеем…), т. е. не было смысла останавливаться на подробном объяснении особенностей работы с каким-либо конкретным Моделлером. О выборе конкретного инструмента – автор высказал свое мнение в тексте.

ЗАКЛЮЧЕНИЕ К ТОМУ II

Настоящий том двухтомника «Теория и практика бизнес-анализа в ИТ. Учебное пособие. Том II» содержит описание основ нотации UML в том объеме, который во многих случаях оказывается достаточным для работы бизнес-аналитика, т. е. модели «Варианты использования» и «Последовательности». Содержатся основы прототипирования интерфейсов.

Как сказал бы автор какого-нибудь романа – «Мы подошли к завершению нашего повествования. К счастью, все трудности позади – наши герои получили все, к чему стремились». Автор этой книги так сказать не может, т. к. эта книга только начало повествования о тернистом пути читателя, который, возможно, решит стать бизнес-аналитиком.

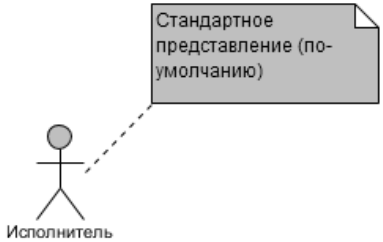
Дальше придется работать самостоятельно: наблюдать, размышлять, пробовать.

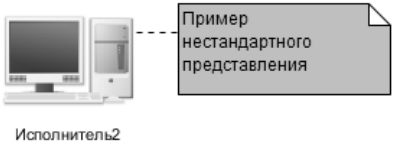
Успехов!

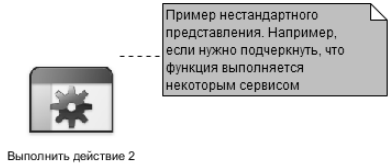
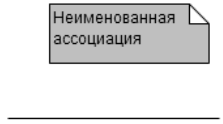
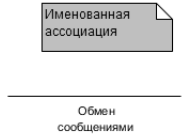
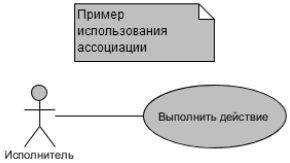
ПРИЛОЖЕНИЕ 1
ЭЛЕМЕНТЫ НОТАЦИИ UML

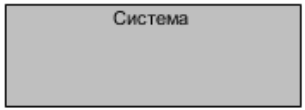
Графические элементы нотации UML, соответствующие документу [3], представлены ниже (см. Таблица 2).

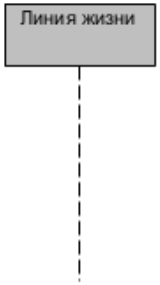
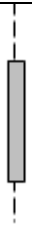
Таблица 2 – Графические элементы нотации UML

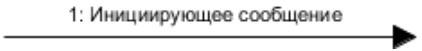
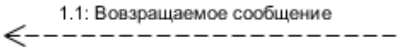
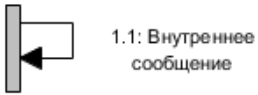
Элемент	Описание	Нотация
Модель (диаграмма) «Варианты использования» (*англ. Use Cases Diagram*)		
Исполнитель (*англ. Actor*)	• Нечто или некто, взаимодействующие с элементом «Вариант использования» (системной функцией); • Наименование элемента «Исполнитель» должно быть именем существительным; • Элемент «Исполнитель» должен выполнять некоторую деятельность, существенную для бизнеса;	

Элемент	Описание	Нотация
	• Элемент «Исполнитель» аналогичен понятию «пользователь», но может иметь несколько ролей (например, сущность «доцент» в реальной жизни может быть и преподавателем, и исследователем); • Элемент «Исполнитель» «запускает» элемент «Вариант использования» (системную функцию); • Элемент «Исполнитель» «воздействует» на систему через входы и «ожидает» реакции системы через выходы	
Вариант использования (*англ. Use Case*)	• Системная функция (процесс может быть автоматическим или ручным); • Наименование элемента «Вариант использования» производится глаголом в сочетании с существительным (или словосочетанием), т. е. обозначает некоторое действие;	

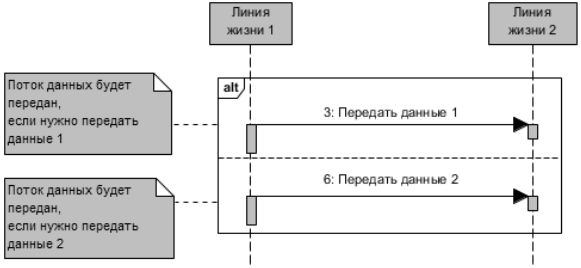
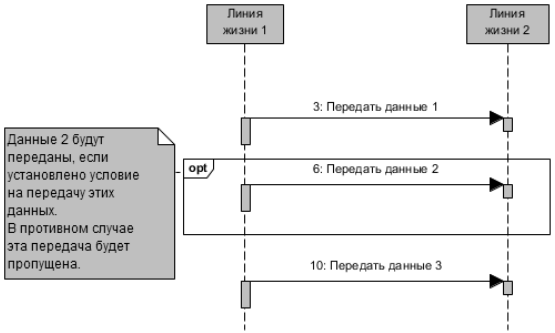
Элемент	Описание	Нотация
	• Каждый элемент «Исполнитель» должен быть связан с элементом «Вариант использования», но не каждый элемент «Вариант использования» соединяется с элементом «Исполнитель»	
Ассоциация (*англ. Association*)	• Участие некоторого Исполнителя во взаимодействии с элементом «Вариант использования». Этот факт отображается путем использования элемента «Ассоциация»; • Демонстрация того, что Исполнитель взаимодействует с «Вариантом использования» через обмен сообщениями	  

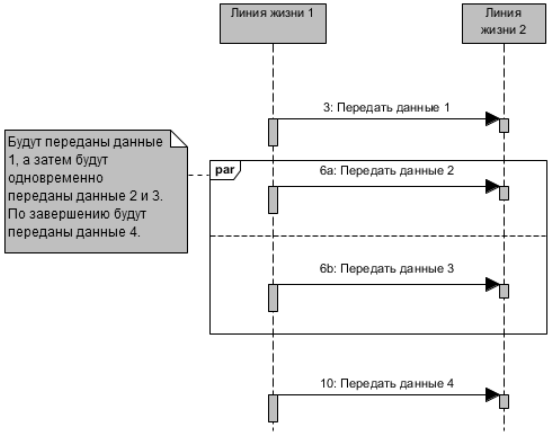
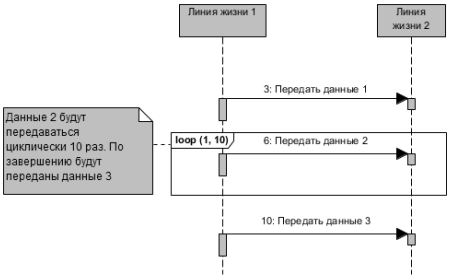
Элемент	Описание	Нотация
Система (*англ. System*)	• Элемент «Система» показывает границы системы целиком так, как это определено в документе с требованиями; • Для больших и сложных систем каждый из модулей может иметь собственные границы, показанные элементом «Система»	
		

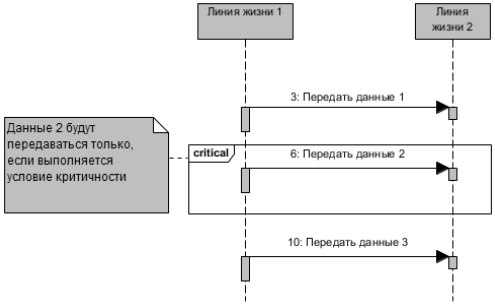
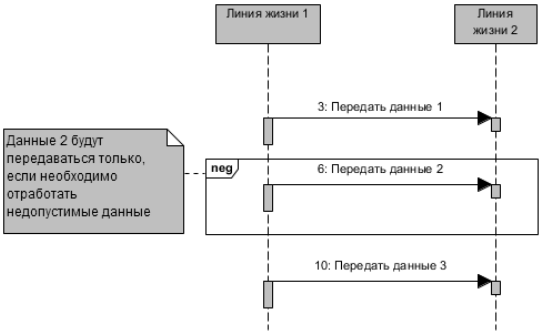
Элемент	Описание	Нотация
Модель (диаграмма) «Последовательности» (*англ. Sequence Diagram*)		
Линия жизни (англ. Lifeline)	Линия жизни изображается с помощью символа, состоящего из прямоугольника, образующего его заголовок, ниже которого следует вертикальная пунктирная линия, отображающая время жизни (продолжительность). Информация, идентифицирующая линию жизни, отображается внутри прямоугольника в следующем формате	
Спецификация выполнения (англ. Activations)	Спецификации исполнения представляют в виде тонких прямоугольников (серого или белого цвета) на линии жизни	

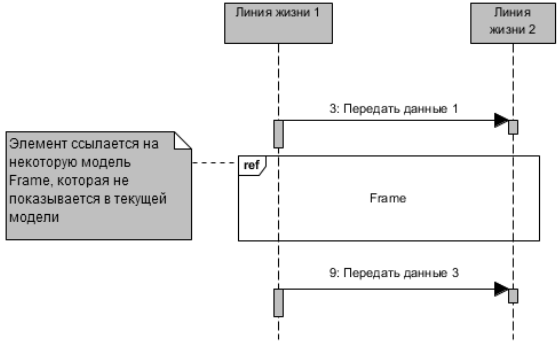
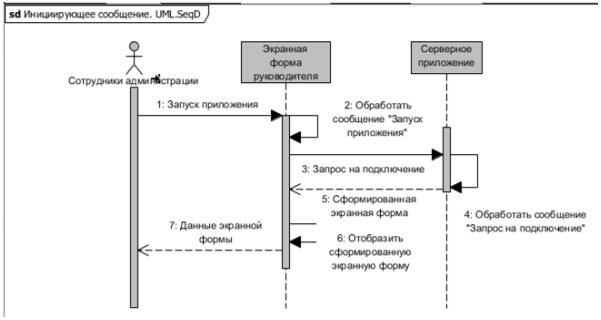
Элемент	Описание	Нотация
Исполнитель (*англ. Actor*)	Элемент «Исполнитель» отображает тип роли, которую играет некоторая сущность, взаимодействующая с заданным субъектом (например, путем обмена сигналами и данными), внешняя по отношению к заданному субъекту (т. е. в том смысле, что экземпляр сущности не является частью экземпляра взаимодействующего с ним субъекта), а также этот элемент представляет роли, которые играют пользователи-люди, внешнее оборудование или другие субъекты. Элемент «Исполнитель» не обязательно представляет конкретную физическую сущность, а просто определяет роль некоторой сущности. Некоторая персона может играть роли нескольких разных исполнителей.	
Инициирующее сообщение (англ. Call Message)	Это сообщение, которое конкретизирует связь между взаимодействующими линиями жизни и вызывает выполнение операции в целевой линии жизни. Инициирующее сообщение может быть именовано так, чтобы пояснять содержание сообщения.	
Возвращаемое сообщение (англ. Return Message)	Это сообщение, которое конкретизирует связь между взаимодействующими линиями жизни и соответствует передаче информации от вызываемой линии жизни к вызывающей линии жизни в соответствии с инициирующим сообщением.	
Внутреннее сообщение (англ. Self Message)	Сообщение, которое адресуется внутри одной линии жизни, т. е. в границах одной линии жизни некоторая операция отправляет сообщение к другой операции, которая относится к этой же линии жизни.	

Элемент	Описание	Нотация
Рекурсивное сообщение (англ. Recursive Message)	Целью рекурсивного сообщения является активация некоторой функции поверх активации, из которой было вызвано сообщение, т. е. функция повторно вызывает сама себя (рекурсивная функция)	2: Рекурсивное сообщение
Порождающее сообщение (англ. Create Message)	Порождающее сообщение предназначено для формирования новой линии жизни, которой не было в исходном состоянии некоторой системы.	3: Порождающее сообщение Линия жизни
Уничтожающее сообщение (англ. Destroy Message)	Сообщение, которое приведет к завершению целевой линии жизни.	4: Уничтожающее сообщение
Длительное сообщение (англ. Duration Message)	Сообщение, которое показывает в явном виде, что от момента отправки сообщения к целевой линии жизни до начала выполнения некоторого процесса в принимающей линии жизни проходит никоторое время. В фигурных скобках указывают продолжительность.	3: Длительное сообщение {Продолжительность}

Элемент	Описание	Нотация
Фрагменты последовательностей (*англ. Sequence Fragments*) – группа элементов, которые облегчают разработку сложных моделей последовательностей. Фрагмент последовательности отображается в виде блока, называемого комбинированным фрагментом, который включает внутри себя часть взаимодействий в модели последовательностей. В левом верхнем углу указывается тип фрагмента последовательностей: ref, assert, loop, break, alt, opt, neg, par (см. ниже в таблице).		
Альтернативные фрагменты последовательностей (*англ. Alternative Sequence Fragments)*	Исполнятся будет только тот фрагмент, для которого задаваемые условия будут истинными. Тип фрагмента последовательностей обозначается, как **alt (см.** Рисунок 25).	 Рисунок 63 – Пример использования альтернативного фрагмента
Возможный фрагмент последовательностей (*англ. Optional Sequence Fragments)*	Фрагмент будет исполнятся только тогда, когда выполняется заданное предварительное условие. Аналогичен альтернативному фрагменту последовательностей, но имеет только один возможный вариант. Тип фрагмента последовательностей обозначается, как **opt**.	

Элемент	Описание	Нотация
Параллельные фрагменты последовательностей (*англ. Parallel Sequence Fragments)*	Фрагменты исполняются параллельно. Тип фрагмента последовательностей обозначается, как **par**.	
Циклический фрагмент последовательностей (*англ. Loop Sequence Fragments)*	Фрагмент исполняется несколько раз до тех пор, пока не будет выполнено условие выхода из цикла. Тип фрагмента последовательностей обозначается, как **loop**.	

Элемент	Описание	Нотация
Фрагмент последовательностей критического участка (*англ. Critical Region Sequence Fragments)*	Фрагмент содержит только один поток, который выполняется только однажды. Тип фрагмента последовательностей обозначается, как **critical**.	
Недопустимый фрагмент последовательностей (*англ. Negative Sequence Fragments)*	Фрагмент содержит поток, который включает в себя недопустимый вывод. Тип фрагмента последовательностей обозначается, как **neg**.	

Элемент	Описание	Нотация
Ссылка на фрагмент последовательностей (*англ. Reference Sequence Fragments*)	Фрагмент является ссылкой на взаимодействие, определенное на другой диаграмме. Рамка нарисована так, чтобы покрыть линии жизни, вовлеченные во взаимодействие. Можно определить параметры и возвращаемое значение. Тип фрагмента последовательностей обозначается, как **ref**.	
Диаграмма последовательностей (*англ. Sequence Diagram*)	Используется для ограничения диаграммы последовательностей. Тип фрагмента последовательностей обозначается, как **sd**.	

СПИСОК ТЕРМИНОВ И СОКРАЩЕНИЙ

БА	Бизнес-анализ
ИБП	Информационный бизнес-процесс
ИС	Информационная система
ИТ	Информационные технологии
НСИ	Нормативно-справочная информация
ПрО	Предметная область
BPMN	Business Process Modeling Language – язык моделирования бизнес-процессов
DLL	Dynamic Link Library, Динамически присоединяемая библиотека (программирование)
ERD	Entity Relationship Diagram или в русской интерпретации: диаграмма сущность-связь
OMG	Международная организация Object Management Group – не-коммерческий консорциум по разработке технологический стандартов. Интернет ресурс: https://www.omg.org/
SeqD	Диаграмма последовательностей
UCD	Диаграмма вариантов использования

СПИСОК ЛИТЕРАТУРЫ

1. Цаленко М. Ш. Моделирование семантики в базах данных. – М.: Наука. Гл. ред. физ.-мат. лит., 1989. – 288 с. – (Проблемы искусственного интеллекта). – ISBN 5-02-014106-2.

2. Буч Г., Рамбо Д., Якобсон Н. «Язык UML. Руководство программиста» // 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2007 – 496 с.: ил.

3. OMG® Unified Modeling Language® (OMG UML®). Version 2.5.1. URL: https://www.omg.org/spec/UML/.

4. OMG Systems Modeling Language ™. Version 1.5. URL: https://www.omg.org/spec/SysML/.

5. Цветков, А. А. Теория и практика бизнес-анализа : учебное пособие. В 2 т. Т. I / А. А. Цветков. – Москва ; Берлин : Директ-Медиа, 2019. – 150 с. ISBN 978-5-4475-8152-7

ОБ АВТОРЕ

Цветков Алексей Анатольевич, научный сотрудник Института программных систем им. А. К. Айламазяна Российской академии наук.

Круг научных интересов: теория и практика разработки ИС, системы искусственного интеллекта, теория сверхбольших баз данных, исследования в области недифференцируемых функций.

Автор многочисленных научных работ в отечественных и зарубежных научных журналах.

Алексей Анатольевич Цветков

Теория и практика бизнес-анализа в ИТ

Том II

Учебное пособие

Ответственный редактор *А. Иванова*
Корректор *М. Глаголева*
Верстальщик *М. Глаголева*

Издательство «Директ-Медиа»
117342, Москва, ул. Обручева, 34/63, стр. 1
Тел/факс + 7 (495) 334–72–11
E-mail: manager@directmedia.ru
www.biblioclub.ru
www.directmedia.ru