Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

М. М. Борисик

УПРАВЛЕНИЕ ИНФОРМАЦИОННЫМИ ПРОЕКТАМИ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве пособия для специальности 1-58 01 01 «Инженерно-психологическое обеспечение информационных технологий»

Минск БГУИР 2019

УДК 004.41(076)
ББК 32.973.202я73
Б82

Р е ц е н з е н т ы:

кафедра интеллектуальных систем
Белорусского национального технического университета
(протокол №2 от 08.10.2018);

главный научный сотрудник государственного научного учреждения
«Институт прикладной физики Национальной академии наук Беларуси»
доктор технических наук А. П. Гусев

Борисик, М. М.

Б82 Управление информационными проектами. Лабораторный практикум : пособие / М. М. Борисик. – Минск : БГУИР, 2019. – 72 с. : ил.
ISBN 978-985-543-502-1.

Пособие посвящено вопросам проектной деятельности, документированию требований к проекту в области информационных технологий, а также основам моделирования и приобретению навыков коммуникации на всех стадиях жизненного цикла проекта. Является методическим обеспечением выполнения лабораторных работ.

УДК 004.41(076)
ББК 32.973.202я73

ISBN 978-985-543-502-1

Содержание

Предисловие

Пособие предназначено для проведения лабораторных работ по дисциплине «Управление информационными проектами» для студентов всех форм обучения специальности 1-58 01 01 «Инженерно-психологическое обеспечение информационных технологий».

В лабораторной работе №1 подробно раскрывается тема содержания проектной деятельности, изучаются основы планирования информационного проекта.

Лабораторная работа №2 посвящена разработке спецификации требований к программному продукту.

В лабораторной работе №3 рассмотрены основы прототипирования.

В лабораторной работе №4 изучается суть визуального моделирования проектов в области информационных технологий.

В лабораторной работе №5 изучается процесс анализа информационного проекта.

Лабораторная работа №6 позволяет приобрести навыки коммуникации на всех стадиях жизненного цикла проекта.

Автор выражает благодарность СООО «Образовательный центр Парка высоких технологий» за предоставленные обучающие материалы.

Лабораторная работа №1
Формирование IT-проекта

Цель: изучить этапы и стадии процесса разработки проектов в области информационных технологий, сформировать знания и навыки работы в программе управления проектами Microsoft Project 2013.

План занятия:

1 Изучить теоретические сведения.

2 Выполнить практическое задание по лабораторной работе.

3 Оформить отчет и защитить лабораторную работу.

Теоретические сведения

Проект – это системный комплекс плановых (финансовых, технологических, организационных и прочих) документов, содержащих комплексно-системную модель действий, направленных на достижение оригинальной цели.

Признаки, которыми должен обладать проект:

1 Наличие цели. Проект всегда направлен на достижение какой-либо конкретной цели, удовлетворение какой-либо потребности. Такое направление предполагает, что есть желаемый измеренный результат, которого можно достичь за определенный срок. Для успешной реализации проекта необходимо не только определить его, но и установить в существенной характеристике, включая условия (требования и ограничения) его функционирования. Цели проекта должны быть четко сформулированными, измеряемыми, ограничения – заданными, а установленные требования – выполнимыми. Цель должна быть определена максимально четко, что позволит оптимально воплотить проект в жизнь.

2 Наличие ресурсной базы. У каждого проекта есть выделенный на него определенный бюджет, и реализация проекта должна осуществляться в рамках этого бюджета. Кроме того, в проекте также присутствуют ограниченные человеческие, материальные и технические ресурсы (техника, оборудование, материалы) – одним словом, оснащение, обеспечивающее выполнение технической стороны проекта.

3 Временные рамки. Длительность осуществления любого проекта лимитируется его начальной (постановка цели и начало затрат усилий на ее достижение) и конечной стадиями (выражается либо в достижении цели, либо в очевидности невозможности ее достижения). Проект не может существовать без ограничений во времени, которые нужны:

- для определения периода, во время которого успех или неудача проекта могут быть очевидны;
- оценки реальной ценности финансовых затрат;
- достижения других целей (необходимость скорейшего выхода на рынок, дата может быть приурочена к запуску коммерческого продукта или услуги, решения необходимы для другого продукта и т. д.).

4 Ожидаемый результат. Проект должен иметь направленность на достижение какого-либо результата, как правило, поставленной цели. От того, насколько полно достигнут ожидаемый результат информационного проекта, зависит успешность самого проекта. Проект также включает координацию выполнения взаимосвязанных действий, поэтому должны быть определены последовательность и объем работ.

Продукт – результат (или набор результатов) поставки по контракту, т. е. то, что хочет получить заказчик.

Проект – это процесс создания продукта, т. е. то, что делает команда, чтобы выдать заказчику продукт. Заказчику обычно проект не интересен.

Управление проектом – это деятельность, направленная на эффективное достижение целей проекта с заданным качеством в установленные сроки в рамках утвержденного бюджета при существующих ограничениях и имеющейся неопределенности за счет использования имеющихся в наличии ресурсов и технологий, а также управленческих знаний, опыта, специализированных подходов, методов и систем.

Необходимо управлять:

1) предметной областью, т. е. реализацией цели;
2) сроками;
3) стоимостью;
4) качеством;
5) рисками;
6) персоналом;
7) коммуникациями;
8) контрактами;
9) изменениями (цели, доступные ресурсы, включая персонал, деньги т. д.).

Проекты имеют определенную дату окончания, бюджет и объем работ. Трио времени, денег и объема часто называют проектным треугольником.

При внесении изменений в один из этих элементов меняются оба других. И хотя для проекта в равной степени важны все три элемента, как правило, только один из них в зависимости от приоритетов имеет наибольшее влияние на другие.

В большинстве проектов по меньшей мере одна сторона треугольника фиксирована, т. е. ее невозможно изменить. Возможно, бюджет не обсуждается или, предположим, продукт непременно должен поступить в продажу к определенной дате. Иногда на руководителя возлагается обязанность определить, какой элемент наиболее важен для успеха проекта. И нужно иметь четкое представление об этом на случай, если возникнут проблемы (а они возникают всегда). Невозможно изменить бюджет, расписание или содержание работ, не повлияв по крайней мере на один из других факторов.

Универсального стандарта качества не существует. Для каждого проекта качество определяется в нем самом. Для некоторых компаний важнейшей мерой качества является соблюдение рамок бюджета. Для других важнее

вовремя вывести продукт на рынок. Руководителю проекта нужно знать, как качество определяется для организации и для самого проекта.

Жизненный цикл проекта – последовательность фаз проекта, задаваемая исходя из потребностей управления проектом. Он имеет пять фаз (рисунок 1.1).

Рисунок 1.1 – Фазы жизненного цикла проекта [1]

1 *Инициация* – принятие решения о начале выполнения проекта. В фазе инициации проекта ведется концептуальное планирование будущего проекта (анализируются проблемы и потребности, собираются исходные данные, определяются цели и задачи проекта, рассматриваются альтернативы), утверждается концепция проекта, открывается проект (принимается решение о начале проекта, определяется и назначается управляющий проектом, принимается решение об обеспечении ресурсами, планируется работа временной рабочей группы проекта).

Работы этой фазы включают мероприятия по маркетингу, подготовке и участию в тендерах и конкурсах и другие мероприятия преддоговорной работы. Горизонт планирования – предполагаемый срок заключения договора (контракта).

2 *Планирование* – определение целей и критериев успеха проекта, разработка рабочих схем их достижения. В данной фазе планируется цель, определяется состав работ проекта, оценивается длительность и объем работ, определяются и назначаются ресурсы проекта, составляется расписание выполнения работ, оценивается бюджет, планируется качество, определяются критерии успеха.

3 *Выполнение* – координация людей и других ресурсов для выполнения плана. Процесс исполнения начинается с момента фиксации базового плана проекта, заканчивается после выполнения обязательств сторон по контракту и является зоной ответственности управляющего проектом.

4 *Контроль* – определение соответствия плана и исполнения проекта поставленным целям и критериям успеха и принятие решений о необходимости применения корректирующих воздействий.

5 *Мониторинг* – определение необходимых корректирующих воздействий, их согласование, утверждение и применение.

6 *Завершение* – формализация выполнения проекта и подведение его к упорядоченному финалу, принятие соответствующего решения руководством компании, анализ и архивация данных проекта.

Жизненный цикл проекта имеет множество процессов, которые распадаются на следующие группы:

1) процессы управления проектами, касающиеся организации и описания работ проекта;

2) процессы, ориентированные на продукт, касающиеся спецификации и производства продукта.

Требования – условие или возможность, необходимые заинтересованному лицу для решения проблемы или достижения цели; характеристики, которые должны быть удовлетворены системой или системным компонентом, чтобы соответствовать контракту, стандарту, спецификации или другому формальному документу.

Можно встретить иные названия этого документа, например: документ бизнес-требований, функциональная спецификация, спецификация продукта, документ о требованиях. Разница в названии не изменяет его назначение. В требованиях указываются функции, возможности, необходимые ограничения, подробное описание поведения системы, необходимое качество системы (производительность, безопасность, удобство использования и т. д.). Однако требования не должны содержать подробности дизайна, проектирования, тестирования и управления проектом, за исключением известных ограничений дизайна и реализации.

Шаблон спецификации требований к программному обеспечению (ПО) представлен на рисунке 1.2.

1 Введение
1.1 Назначение
1.2 Соглашения, принятые в документах
1.3 Границы проекта
1.4 Ссылки
2 Общее описание
2.1 Общий взгляд на продукт
2.2 Классы и характеристики пользователей
2.3 Операционная среда
2.4 Ограничения дизайна и реализации
2.5 Предположения и зависимости
3 Функции системы
3.x Функция системы X
3.x.1 Описание
3.x.2 Функциональные требования
4 Требования к данным
4.1 Логическая модель данных
4.2 Словарь данных
4.3 Отчеты
4.4 Получение, ценность, хранение и утилизация данных
5 Требования к внешним интерфейсам
5.1 Пользовательские интерфейсы
5.2 Интерфейсы ПО
5.3 Интерфейсы оборудования
5.4 Коммуникационные интерфейсы
6 Атрибуты качества
6.1 Удобство использования
6.2 Производительность
6.3 Безопасность
6.4 Техника безопасности
6.x [Другие]
7 Требования по интернационализации и локализации
8 Остальные требования
Приложение А. Словарь терминов
Приложение Б. Модель анализа

Рисунок 1.2 – Шаблон для спецификации требований к ПО [2]

1 ВВЕДЕНИЕ

Введение представляет собой обзор, помогающий разобраться в структуре и принципе использования спецификации требований к ПО.

1.1 Назначение

Определение продукта или приложения, требования для которого указаны в этом документе, в том числе редакции или номера выпуска. Если спецификация требований к ПО относится только к части системы, то идентифицируют эту часть или подсистему. Описываются типы лиц, которым адресован этот документ.

1.2 Соглашения, принятые в документах

Описываются все стандарты или типографические соглашения, включая значение стилей текста, особенности выделения или нотацию.

1.3 Границы проекта

Кратко описывается ПО и его назначение. Показывается, как связан продукт с пользователями или корпоративными целями, а также с бизнес-целями и стратегиями. Если имеется отдельный документ о концепции и границах проекта, его содержимое не повторяется, а просто ссылаются на него.

1.4 Ссылки

Перечисляются все документы или другие ресурсы, на которые ссылаются в этой спецификации, в том числе гиперссылки на них, если их местоположение меняться не будет. Объем информации должен быть достаточным для того, чтобы пользователь сумел при необходимости получить доступ к каждому указанному материалу, а именно: название, имя автора, номер версии, дата, источник, место хранения или URL-адрес.

2 ОБЩЕЕ ОПИСАНИЕ

В этом разделе представлен общий обзор продукта и среды, в которой он будет применяться, предполагаемая пользовательская аудитория, а также известные ограничения, предположения и зависимости.

2.1 Общий взгляд на продукт

Описывается контекст и происхождение продукта. Поясняется, является он новым членом растущего семейства продуктов, новой версией существующей системы, заменой существующего приложения или совершенно новым продуктом.

2.2 Классы и характеристики пользователей

Определяются различные классы пользователей, которые, как предполагается, будут работать с вашим продуктом, и описываются их соответствующие характеристики.

2.3 Операционная среда

Описывается рабочая среда, в которой будет работать ПО, включая аппаратную платформу, операционные системы и их версии, а также географическое местоположение пользователей, серверов и баз данных вместе с организациями, в которых располагаются соответствующие базы данных, серверы и вебсайты.

2.4 Ограничения дизайна и реализации

Описываются все факторы, которые ограничат доступные разработчикам возможности, например: использование вполне определенного языка программирования, определенной библиотеки, на разработку которой уже потрачено время, и т. п., и логически обосновывается каждое положение.

2.5 Предположения и зависимости

Включаемые здесь предположения относятся к системной функциональности; предположения, относящиеся к бизнесу, представляются в документе концепции и границ проекта. Определяются все зависимости проекта или создаваемой системы от внешних факторов или компонентов вне ее контроля.

3 ФУНКЦИИ СИСТЕМЫ

3.x Функция системы X

Описывается название особенности несколькими словами, например «3.1 Проверка правописания». Так же называются подразделы с 3.x.1 по 3.x.3 для каждой функции системы.

3.x.1 Описание

Кратко описывается функция системы и указывается ее приоритет (они могут изменяться в ходе проекта). Если используется средство управления требованиями, определяется атрибут требований для обозначения приоритета.

3.x.2 Функциональные требования

Перечисляются по пунктам конкретные функциональные требования, которые связаны с этой функцией. Именно эти функции ПО нужно реализовать, чтобы пользователь мог использовать сервисы этой функции или реализовать вариант использования. Описывается, как продукт должен реагировать на ожидаемые ошибки – неправильный ввод информации или неверные действия.

4 ТРЕБОВАНИЯ К ДАННЫМ

Описываются различные аспекты данных, которые будет потреблять система в качестве входной информации, как-то обрабатывать и возвращать в виде выходной информации.

4.1 Логическая модель данных

Это визуальное представление объектов и наборов данных, которые будет обрабатывать система, а также отношений между ними. Существует много видов нотации для моделирования данных, в том числе диаграммы отношений «сущность – связь» и диаграммы классов UML.

4.2 Словарь данных

Словарь данных лучше хранить как отдельный артефакт, не внедряя его в спецификацию требований к ПО (повышает возможности повторного использования в других проектах).

4.3 Отчеты

Перечисляются и описываются характеристики отчетов, если они будут генерироваться приложением. Отчет должен соответствовать определенному готовому макету.

4.4 Получение, целостность, хранение и утилизация данных

Описывается, как получают и обслуживают данные.

5 ТРЕБОВАНИЯ К ВНЕШНИМ ИНТЕРФЕЙСАМ

5.1 Пользовательские интерфейсы

Описываются логические характеристики каждого пользовательского интерфейса.

Пример: стандарты шрифтов, названий кнопок, стандарты отображения текста сообщений, стандарты конфигурации интерфейса для упрощения локализации ПО, специальные возможности для пользователей с проблемами со зрением и т. д.

5.2 Интерфейсы ПО

Описываются связи продукта и других компонентов ПО (идентифицированные по имени и версии), в том числе другие приложения, базы данных, операционные системы, средства, библиотеки, веб-сайты и интегрированные серийные компоненты.

5.3 Интерфейсы оборудования

Описываются характеристики каждого интерфейса между компонентами ПО и оборудования системы.

5.4 Коммуникационные интерфейсы

Указываются требования для любых функций взаимодействия, которые будут использоваться продуктом, включая электронную почту, веб-браузер, сетевые протоколы и электронные формы.

6 АТРИБУТЫ КАЧЕСТВА

Указываются относительные приоритеты различных атрибутов.

6.1 Удобство использования

Требования к удобству использования подразумевают легкость изучения, простоту использования.

6.2 Производительность

Указываются конкретные требования к производительности для различных системных операций.

6.3 Безопасность

Указываются все требования, касающиеся безопасности или конфиденциальности, которые ограничивают доступ или возможности использования продукта.

6.4 Техника безопасности

В этом разделе указываются требования, связанные с возможными потерями, повреждениями или ущербом, которые могут быть результатом использования продукта.

7 ТРЕБОВАНИЯ ПО ИНТЕРНАЦИОНАЛИЗАЦИИ И ЛОКАЛИЗАЦИИ

Требования по интернационализации и локализации обеспечивают возможность использовать продукт в других странах, региональных стандартах и географических районах, отличающихся от тех, в которых он был создан.

8 ОСТАЛЬНЫЕ ТРЕБОВАНИЯ

Определяются все другие требования, которые еще не были описаны в спецификации требований к ПО.

Приложение А. Словарь терминов

Определяются все специальные термины, которые необходимо знать для правильного понимания спецификации требований к ПО.

Работа по сбору требований, их оценке, структуризации и документировании – самый важный этап в процессе разработке ПО. Правильно определенные и задокументированные, они позволяют быть уверенными в том, что команда будет работать над тем, что нужно заказчику, что приоритеты расставлены правильно, что вы знаете, какие ресурсы и на каком этапе вам понадобятся.

Процесс сбора требований можно разделить на четыре этапа (рисунок 1.3):

1) выявление;
2) анализ;
3) спецификация (документирование результатов);
4) валидация (согласование).

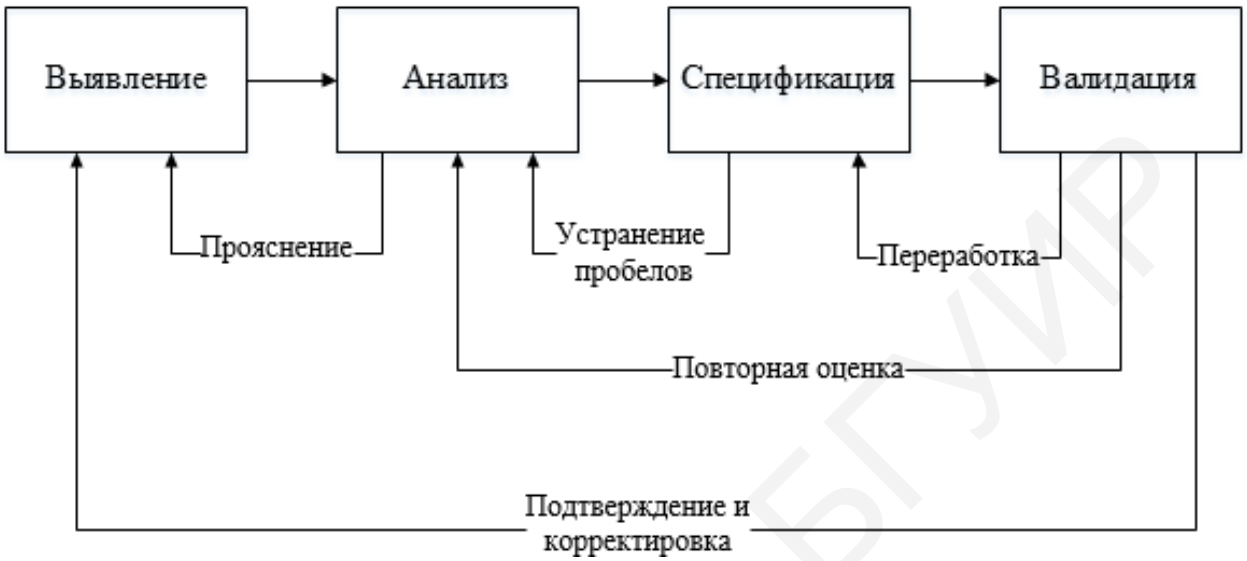


Рисунок 1.3 – Процесс сбора требований [1]

1 Выявление:

а) *интервью*;

б) *анкетирование*;

в) *воркшопы по сбору требований.* Такие семинары позволяют развиться более тесному сотрудничеству аналитиков и клиентов, являются прекрасным способом выявить нужды пользователей и составить наброски документов с требованиями;

г) *наблюдение за пользователями на их рабочих местах.* Диаграмма рабочего процесса позволяет выяснить, где и какие данные были задействованы, а каких не хватает. Такой анализ помогает выявить потенциал развития существующей системы или применения нового продукта;

д) *анализ существующих систем (в том числе конкурентных)* необходим в случае, когда участники процесса не уверены, какие дополнительные данные и из каких источников им могут понадобиться. Например, при настройке программы управления кадрами необходимо достоверно знать, какие данные требуются к передаче от отдела кадров в бухгалтерию в момент приема и/или увольнения сотрудника;

е) *анализ документации.* Зачастую при анализе документации обнаруживаются ограничения, существующие в организации, но не озвученные пользователями. А иногда бывает так, что на бумаге существует гораздо более простое и логичное решение, когда-то сотворенное, но не внедренное.

2 Анализ. Информация, полученная от пользователей, изучается, отделяются задачи от функциональных и нефункциональных требований, бизнес-правил, предполагаемых решений и другой информации. Анализ подразумевает:

а) разложение высокоуровневых требований до нужного уровня детализации;

б) выведение функциональных требований из информации других требований;

в) понимание относительной важности атрибутов качества;

г) распределение требований по компонентам ПО, определенным в системной архитектуре;

д) согласование приоритетов реализации;

е) выявление пробелов в требованиях или излишних требований, не соответствующих заданным рамкам.

3 Спецификация. Результатом анализа должны быть следующие документы: концепция системы, спецификация вариантов использования, спецификация требований, прототипы интерфейса.

Содержание *концепции системы* (Vision and Scope Document):

1) бизнес-требования:

а) исходные данные;

б) возможности бизнеса;

в) бизнес-цели и метрики успешности;

г) изложение концепции;

д) бизнес-риски;

е) бизнес-допущения и зависимости;

2) рамки проекта и ограничения:

а) основные функции;

б) состав первой версии системы;

в) состав последующих версий;

г) ограничения и исключения;

3) бизнес-контекст:

а) участники проекта;

б) приоритеты;

в) аспекты развертывания.

Содержание *спецификации вариантов использования* (Use Case Document):

1) классы пользователей;

2) список вариантов использования;

3) диаграммы вариантов использования;

4) описание каждого варианта использования.

Содержание *спецификации требований* (Software Requirement Specification):

1) общая информация;

2) функциональные требования;

3) нефункциональные требования.

4 Валидация. Сам аналитик, команда разработки, клиент, команда тестирования вместе и/или по отдельности изучают составленные на предыдущем этапе документы, чтобы исключить неправильное описание требований.

Завершать процесс сбора требований необходимо в следующих случаях:
1) новые требования не появляются;
2) пользователи повторно обсуждают упоминавшиеся проблемы;
3) новые требования появляются, но с низким приоритетом;
4) новые требования появляются, но для будущих релизов;
5) предлагаемые требования выходят за границы проекта.

Создание проекта в Microsoft Project 2013

Для создания нового проекта нужно выполнить предварительную настройку. Для уже открытого проекта (рисунок 1.4) в меню *Файл* следует выбрать *Параметры*.

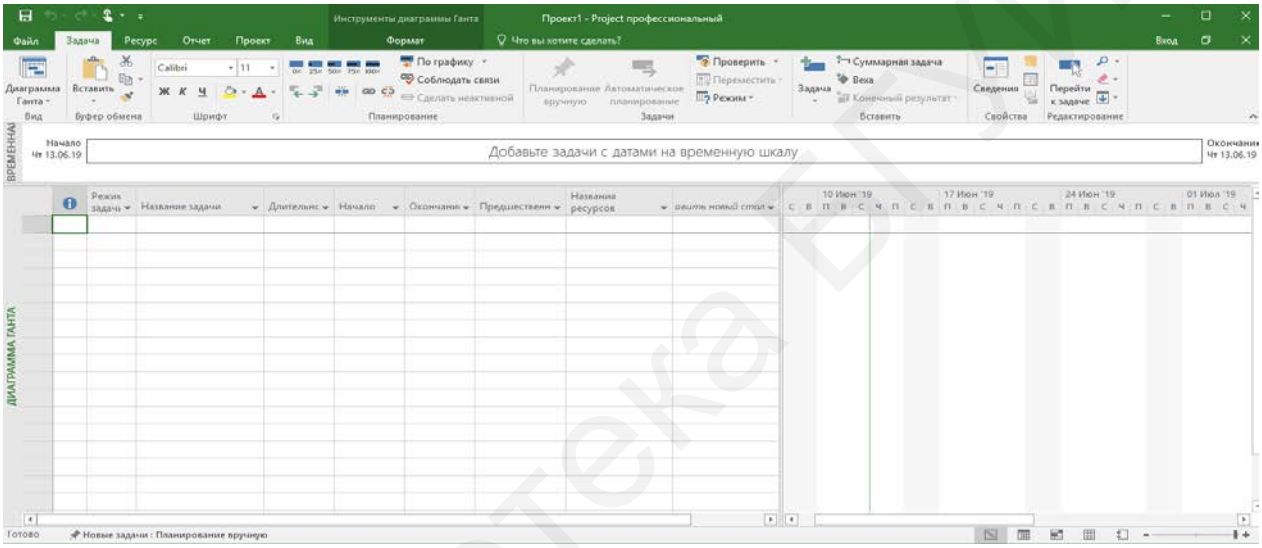


Рисунок 1.4 – Внешний вид рабочей среды Microsoft Project 2013

В появившемся окне необходимо изменить заполнение полей. В поле *Планируемые задачи* необходимо указать «Планирование автоматически» (рисунок 1.5). В поле *Фиксированный объем работ* изменить объем работ на «Фиксированная длительность» (рисунок 1.6).

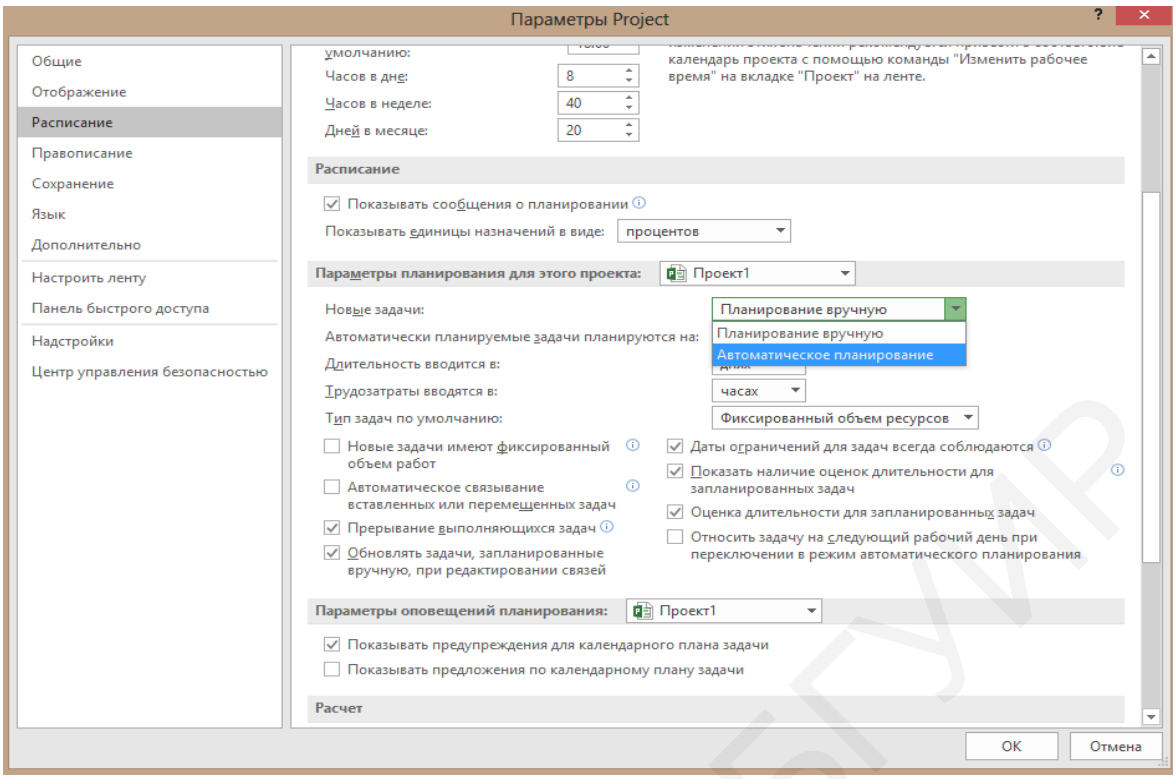


Рисунок 1.5 – Предварительная настройка поля *Планируемые задачи*

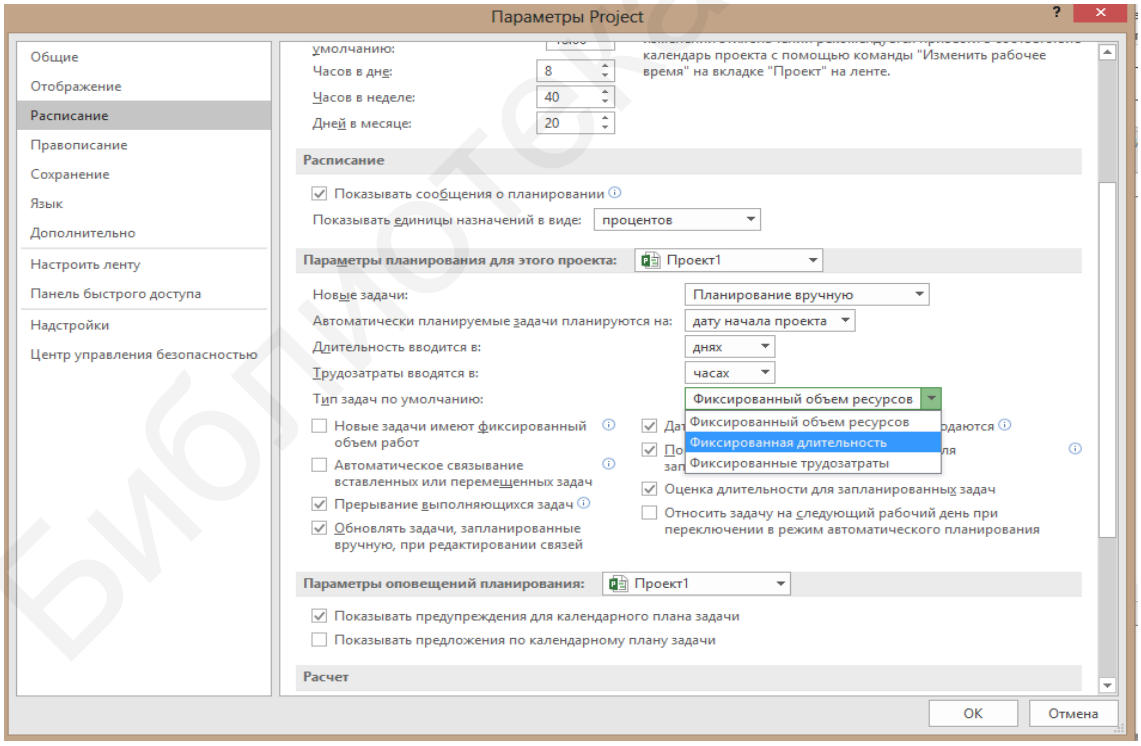


Рисунок 1.6 – Предварительная настройка поля *Фиксированный объем работ*

Находящаяся слева панель представления содержит переключатель рабочих пространств/сред (рисунок 1.7). Панель можно вызвать, щелкнув правой кнопкой мыши в левой части программы (рисунок 1.8).

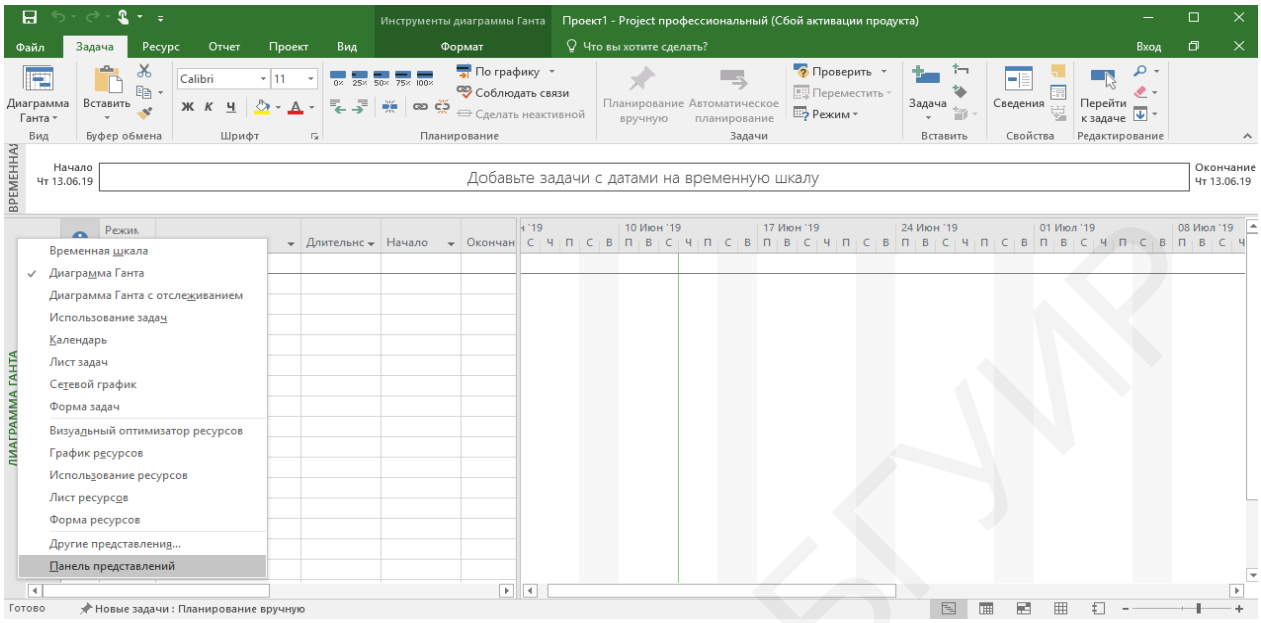


Рисунок 1.7 – Вызов панели представления

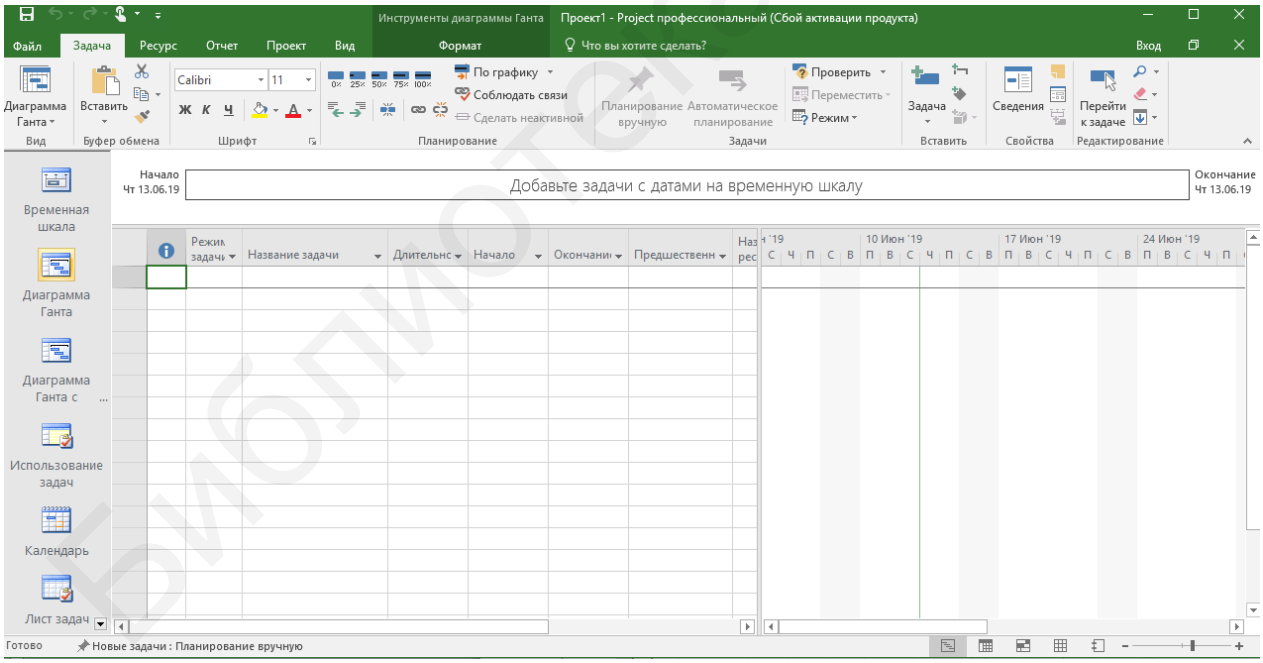


Рисунок 1.8 – Внешний вид панели представления

Панель представлений содержит значки с названиями представлений, щелкая на которых можно быстро переключаться между различными представлениями. Например, на рисунке 1.9 видно, что после щелчка на значке представления *Назначение вех* в окне отображается представление.

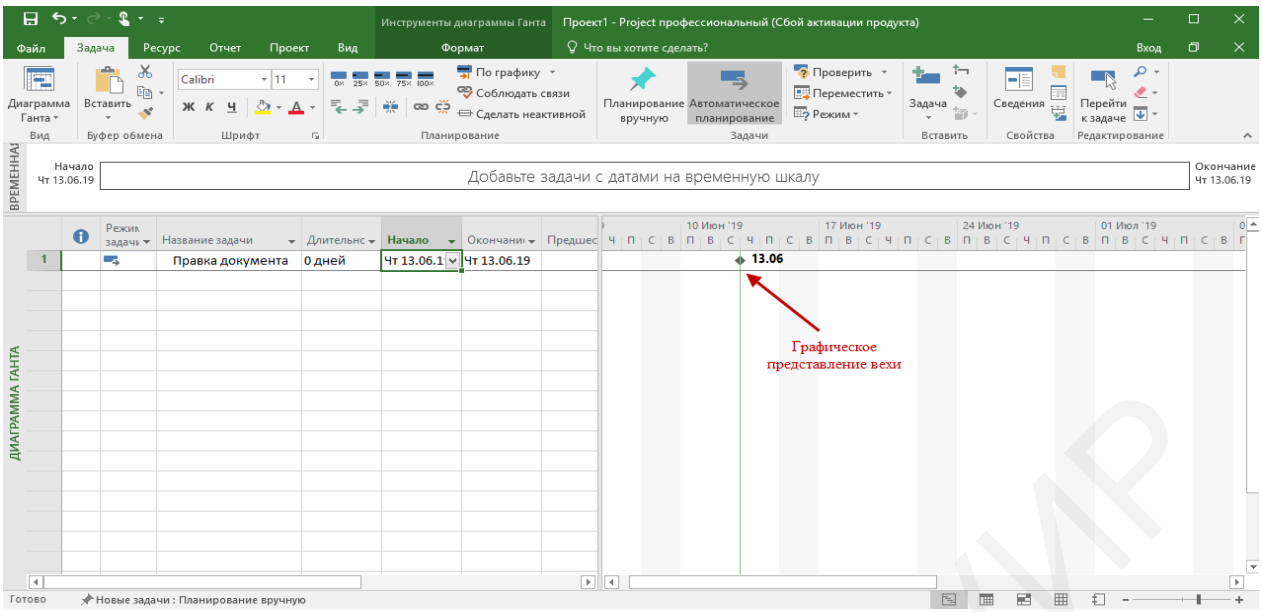


Рисунок 1.9 – Графическое представление вехи

Веха – это работа с нулевой длительностью (используется при создании расписания проекта, чтобы обозначить значимый этап, событие или дату в ходе осуществления проекта). Веха используется для отображения состояния завершенности тех или иных работ. В контексте проекта руководители используют вехи для того, чтобы обозначить важные промежуточные результаты, которые должны быть достигнуты в процессе реализации проекта. Последовательность вех, определенных руководителем, часто называется планом по вехам. Даты достижения соответствующих вех образуют календарный план по вехам.

Все необходимые команды располагаются на ленте (рисунок 1.10).

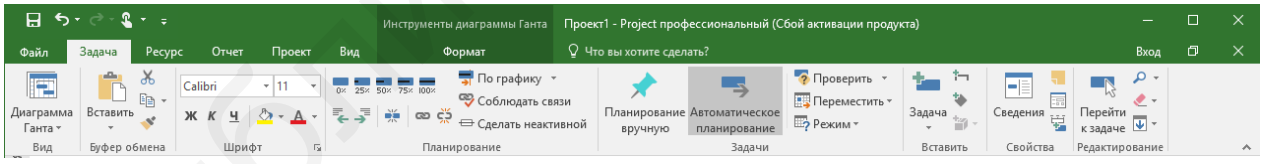


Рисунок 1.10 – Лента интерфейса Microsoft Project 2013

Команды на ленте распределены по вкладкам *Задача*, *Ресурс*, *Отчет*, *Проект*, *Вид* и *Формат* и упорядочены в логические группы, что помогает быстро находить требуемую кнопку для выполнения необходимого действия, например *Назначение связей* (рисунок 1.11).

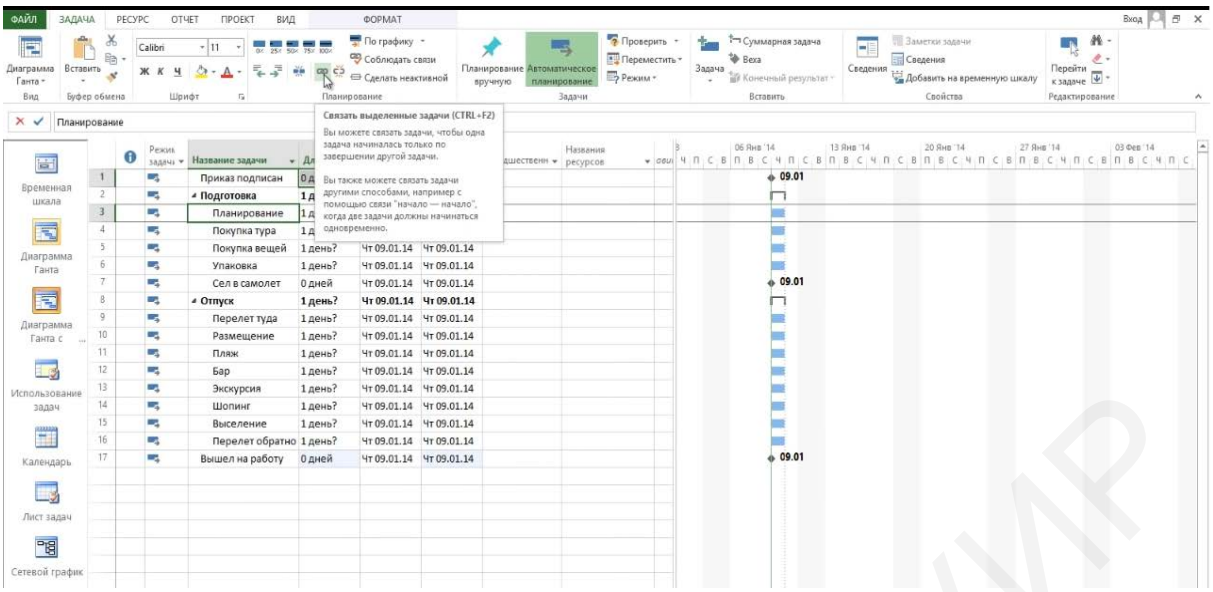


Рисунок 1.11 – Применение функции *Назначение связей*

Связи предшествования (логические зависимости) отображают природу зависимостей между работами. Большинство связей в проектах относятся к типу «конец – начало», когда последующая работа может начаться только по завершении предшествующей работы. Связи предшествования образуют структуру сети. Комплекс взаимосвязей между работами часто также называют логической структурой проекта, поскольку он определяет последовательность выполнения работ.

Для планирования базовой версии проекта следует выполнить следующие шаги (рисунок 1.12):

1) планирование целей – разработка постановки задачи (проектное обоснование, основные этапы и цели проекта);

2) декомпозиция целей – декомпозиция этапов проекта на более мелкие и более управляемые компоненты для обеспечения более действенного контроля;

3) определение состава операций (работ) проекта – составление перечня операций, из которых состоит выполнение различных этапов проекта;

4) определение взаимосвязей операций – составление и документирование технологических взаимосвязей между операциями;

5) оценка длительностей или объемов работ – оценка количества рабочих временных интервалов либо объемов работ, необходимых для завершения отдельных операций;

6) определение ресурсов (людей, оборудования, материалов) проекта – определение общего количества ресурсов всех видов, которые могут быть использованы на работах проекта (ресурсов организации), и их характеристик;

7) назначение ресурсов – определение ресурсов, необходимых для выполнения отдельных операций проекта;

8) оценка стоимостей – определение составляющих стоимостей операций проекта и оценка этих составляющих для каждой операции, ресурса и назначения;

9) составление расписания выполнения работ – определение последовательности выполнения работ проекта, длительностей операций и распределения во времени потребностей в ресурсах и затрат с учетом наложенных ограничений и взаимосвязей;

10) оценка бюджета – приложение оценок стоимости к отдельным компонентам проекта (этапам, фазам, срокам);

11) разработка плана исполнения проекта – интеграция результатов остальных подпроцессов для составления полного документа;

12) определение критериев успеха – разработка критериев оценки исполнения проекта.

Рисунок 1.12 – Основные этапы планирования проекта [3]

Кроме основных процессов есть еще и вспомогательные процессы (рисунок 1.13).

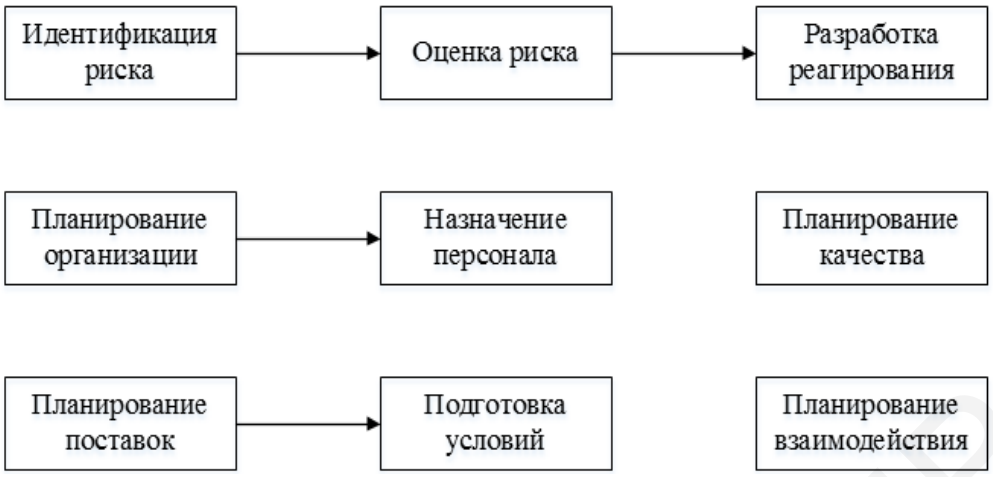


Рисунок 1.13 – Вспомогательные процессы планирования проекта [3]

Вспомогательные процессы планирования:

1) планирование качества – определение того, какие стандарты качества использовать в проекте, и того, как эти стандарты достичь;

2) планирование организации – определение, документирование и назначение ролей, ответственности и взаимоотношений отчетности в организации;

3) назначение персонала – назначение человеческих ресурсов на выполнение работ проекта;

4) планирование взаимодействия – определение потоков информации и способов взаимодействия, необходимых для участников проекта;

5) идентификация риска – определение и документирование событий риска, которые могут повлиять на проект;

6) оценка риска – оценка вероятностей наступления событий риска, их характеристик и влияния на проект;

7) разработка реагирования – определение необходимых действий для предупреждения рисков и реакции на угрожающие события;

8) планирование поставок – определение того, что, как и когда должно быть поставлено;

9) подготовка условий – выработка требований к поставкам и определение потенциальных поставщиков.

Взаимосвязи между вспомогательными подпроцессами, как и само их наличие, в большей мере зависят от природы проекта.

В фазе планирования проекта определяются соисполнители этапов и работ проекта, и формируется базовый план. Горизонт планирования – предполагаемый срок окончания проекта.

Практическое задание:

1 Разработать описание технического проекта, определенного для дальнейшего развития в лабораторной работе. В разработанном описании отразить название проекта; описать проблему, которую будет решать проект; выполнить описание потенциальных пользователей разработки; разработать перечень

предложений для заинтересованных лиц (пользователей проекта); выполнить описание технологий, которые будут применены при выполнении проекта.

2 Разработать перечень функций проекта (определить не менее 5 функций пользователя и не менее 10 функций системы).

3 Выполнить декомпозицию этапов проекта на более мелкие и более управляемые компоненты.

4 Разработать перечень операций для выполнения различных этапов проекта.

5 Разработать контрольные точки проекта.

6 Разработать последовательность выполнения работ.

7 Разработать графическую визуализацию проекта с использованием программы управления проектами Microsoft Project 2013.

8 Обеспечить набор команды (4 чел.) для выполнения проекта. Привести характеристику сформированной группы: осуществить описание личностных качеств каждого члена (все, что может повлиять на успешность выполнения проекта), указать приобретенные профессиональные качества каждого участника, описать другую релевантную информацию.

9 Проанализировать результаты. Оформить отчет и ответить на контрольные вопросы.

Содержание отчета:

1 Цель работы.

2 Формулировка идеи проекта в единое предложение.

3 Перечень функций проекта.

4 Скриншоты графического представления проекта в Microsoft Project 2013.

5 Выводы по работе.

Контрольные вопросы:

1 Какие функциональные возможности есть у программы Microsoft Project 2013?

2 Какие существуют фазы жизненного цикла проекта?

3 Какие существуют основные и вспомогательные процессы этапа планирования?

4 Какие существуют основные и вспомогательные процессы этапа исполнения?

5 Какие существуют основные и вспомогательные процессы этапа контроля?

6 Какие существуют основные и вспомогательные процессы этапа завершения?

7 Какие существуют признаки проекта? Дайте характеристику каждому признаку.

8 Назовите этапы сбора требований.

9 Какими характеристиками обладает этап сбора требований «Выявление»?

10 Какими характеристиками обладает этап сбора требований «Анализ»?

11 Какими характеристиками обладает этап сбора требований «Спецификация»?

Лабораторная работа №2
Документирование требований

Цель: изучить категории требований, используя стандарты и шаблон спецификации требований, научиться составлять спецификацию требований.

План занятия:
1 Изучить теоретические сведения.
2 Выполнить практическое задание по лабораторной работе.
3 Оформить отчет и ответить на контрольные вопросы.

Теоретические сведения

Понимание того, что нужно для работы с требованиями, начинается с осознания потребности в зрелом процессе управления качеством требований (рисунок 2.1).

Рисунок 2.1 – Уровни процесса управления качеством требований

УРОВЕНЬ 0 – ОТСУТСТВИЕ ТРЕБОВАНИЙ

Нулевой уровень зрелости по умолчанию может быть присвоен любой команде, которая разрабатывает программное обеспечение. Отсутствие требований к программному обеспечению проявляется в следующих случаях:

- команда разработчиков не встречается с заказчиком для получения требований, потому что они уверены в том, какой продукт им необходимо реализовать;
- члены команды считают, что экономят проектное время, пропуская задачи выявления и документирования требований.

В результате требования не документируются, появляются споры и непонимание между членами команды. При смене членов команды требования могут быть потеряны, т. к. человек, который их помнил, уволился. Игнорирование стадии выявления требований приводит к тому, что программный продукт не удовлетворяет потребностям заказчика, потому что в нем отсутствует необходимый функционал или, наоборот, реализованы функции, которые заказчику не нужны.

УРОВЕНЬ 1 – ДОКУМЕНТИРОВАНИЕ ТРЕБОВАНИЙ

Цель первого уровня зрелости заключается в получении документов, описывающих требования. Наличие документов с требованиями является отправной точкой для процесса управления требованиями, поскольку они являются базисом для дальнейшей разработки программного обеспечения. Наличие документов также сокращает риски, связанные со сменой персонала и потерей требований.

Для того чтобы проверить качество подобных спецификаций, рекомендуется использовать проверку документа экспертом предметной области или заказчиком.

УРОВЕНЬ 2 – ОРГАНИЗАЦИЯ ТРЕБОВАНИЙ

Целью второго уровня зрелости является получение набора требований, понятных и заказчику, и членам проектной команды. Требования должны понятно, непротиворечиво и однозначно описывать желаемое поведение разрабатываемой системы. Для того чтобы получить требования с такими критериями качества, необходимо разработать шаблоны документов, собрать требования в едином документе/хранилище и вести историю изменений. Документы должны полностью соответствовать всем предъявляемым к нему требованиям по оформлению. Так как требования могут использоваться различными членами проектной команды и заказчиком, необходимо иметь возможность разграничения доступа к требованиям.

Первым шагом к анализу требований является поиск противоречивых и неполных требований. В результате проверки требований или в момент

их документирования, могут возникнуть конфликты между требованиями, что приведет к необходимости уточнения требований. В таком случае необходимо снова вернуться к выявлению требований, чтобы заполнить пробелы в информации и исправить возникшие противоречия.

В проверке качества требований участвуют не только аналитик и эксперт предметной области, а также другие члены проектной команды. Системный архитектор или ведущий разработчик проверяют требования на возможность их реализации с использованием принятых в команде технологий, платформ и других ограничений, инженеры по тестированию проверяют требования на возможность их дальнейшего тестирования. Подобные проверки способствуют достижению таких критериев качества, как полнота и проверяемость.

УРОВЕНЬ 3 – СТРУКТУРИРОВАНИЕ ТРЕБОВАНИЙ

Цель третьего уровня зрелости заключается в необходимости планирования процесса управления качеством требований, разделения требований на типы по объединяющим признакам.

Разделение требований по типам позволит классифицировать огромное множество требований, структурировать их по общим признакам, что позволит членам проектной команды лучше ориентироваться в большом количестве требований и управлять ими. Структуризация требований также позволит выявить повторяющиеся и противоречивые требования, обеспечивая тем самым соответствующие критерии качества.

Также данный уровень добавляет необходимую информацию к требованиям, предусматривает добавление атрибутов к требованиям, применение моделей при анализе и документировании требований, которое позволяет решить проблемы понимания требований и дальнейшего моделирования и проектирования программной системы. При использовании моделей для анализа требований возможен плавный и последовательный переход к моделям проектирования.

УРОВЕНЬ 4 – ТРАССИРОВКА ТРЕБОВАНИЙ

Достижение предыдущих трех уровней зрелости приведет проектную команду к тому, что можно будет определять и отслеживать отношения между требованиями.

Целью установки отношений между требованиями (трассировки) является получение возможности отслеживания изменений требований, их влияния друг на друга, возможность идентификации избыточных и неучтенных требований. Установление подобных связей совершенствует процесс управления требованиями посредством применения строгих правил отношения, анализа полноты учета и анализа влияния.

УРОВЕНЬ 5 – КОМПЛЕКСНОСТЬ ТРЕБОВАНИЙ

Пятый уровень зрелости нацелен на использование требований не только для согласования с заказчиком, но и для дальнейшей разработки программного обеспечения.

Достижение пятого уровня позволяет инструментальным средствам, предназначенным для работы с требованиями, интегрироваться с остальной частью окружающей среды разработки программного обеспечения. Это означает использование требований непосредственно в проектировании программного обеспечения, управлении изменениями, тестировании и руководстве проектом. Для достижения подобных целей необходимо разработать и внедрить в процесс управления требованиями показатели их оценки для управления проектом, осуществить трассировку требований на элементы проектирования и тестирования, а также внедрить систему управления изменениями.

Одним из основных преимуществ документирования требований, нацеленных на получение качественного продукта, является согласование документов с заинтересованными лицами.

Классификация требований представлена на рисунке 2.2.

Рисунок 2.2 – Категории требований

Существует три уровня функциональных требований:

1) бизнес-требования;
2) требования пользователей;
3) функциональные требования.

Бизнес-требования – высокоуровневые требования, которые содержат бизнес-цели организации или заказчика системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга или «провидец» продукта.

Отвечают на вопрос: что мы делаем и зачем?

Источник: заказчик, спонсор проекта.

Пример: разработать сайт интернет-магазина по торговле автозапчастями с целью увеличения объема продаж на 20 % в течение 12 мес. после запуска.

Пользовательские требования описывают потребности пользователей системы, а также способы взаимодействия пользователей и системы.

Основным подходом к описанию пользовательских требований является создание вариантов использования. Они описывают последовательность взаимодействия системы и внешнего действующего лица. Действующим лицом может быть человек, другая система ПО или аппаратное устройство, взаимодействующее с системой для достижения некой цели.

Отвечают на вопрос: что хотят от системы пользователи и заинтересованные лица?

Источник: пользователи и заинтересованные лица.

Пример: возможность совершения онлайн-заказа в интернет-магазине.

Функциональные требования – положения, которые описывают функциональные возможности или поведение системы при определенных условиях. Они определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований.

Отвечают на вопрос: что необходимо реализовать в системе?

Источник: аналитики.

Пример: система должна отображать в табличном виде страницу «Список заказов», содержащую все поступившие за день заказы, с возможностью сортировки по времени поступления.

Функциональное требование строится по формуле «ID → Условие → Система должна → Действие». При этом ID включает аббревиатуру – сокращение от названия элемента (обычно на английском языке) и порядковый номер с учетом иерархии. Для описания функциональных требований может быть использован следующий шаблон (таблица 2.1).

Таблица 2.1 – Описание функциональных требований

ID	Название функции	Роли	Описание	Функциональные требования	Приоритет	Связанные элементы
F1	Создание аккаунта	Гость	Позволяет создать новый аккаунт	FR1-1. Система позволяет ввести данные аккаунта. FR1-2. Система позволяет проверить корректность введенных данных. FR1-3. Система позволяет создать аккаунт с помощью социальной сети. FR1-4. Система позволяет отметить создание нового аккаунта	Средний	F2
F2	…	…	…	FR2-1… FR2-2…	…	…

Нефункциональное требование – описание свойств или характеристик, которые система должна демонстрировать, или ограничения, которые она должна соблюдать.

Выделяют четыре уровня нефункциональных требований:

1) бизнес-правила;
2) атрибуты качества;
3) ограничения;
4) внешние интерфейсы.

Бизнес-правила – это положение, определяющее или ограничивающее какие-либо стороны бизнеса, его назначение – защитить структуру бизнеса, контролировать или влиять на его операции. Бизнес-правила могут регулировать деятельность организации как снаружи (законодательство, стандарты отрасли и т. п.), так и внутри (корпоративные политики, положения о подразделениях, должностные инструкции и т. п.). Иногда бизнес-правила становятся источником атрибутов качества, которые реализуются в функциональности. Важно понимать, что внутренние бизнес-правила не всегда задокументированы и могут быть в головах лишь у конкретных сотрудников.

Пример: магазин обрабатывает заказы покупателей только до 20:00 текущего дня.

Для документирования бизнес-правил можно использовать следующий шаблон (таблица 2.2).

Таблица 2.2 – Шаблон для документирования бизнес-правил

ID	Описание	Тип	Статическое или динамическое	Источник
F1	Позволяет создать новый аккаунт			

Атрибуты качества – требования, которые устанавливают параметры, которые будут использоваться для оценки работы системы.

Сюда относятся:

а) надежность – система работает тогда, когда нужно;

б) удобство – каждому пользователю удобно и просто использовать систему для решения задач;

в) ремонтопригодность – легко ли исправить или заменить систему;

г) масштабируемость – может ли система обрабатывать больше данных или обслуживать большее количество пользователей без переделки;

д) переносимость – система работает на разных операционных системах, в разных средах, на разных устройствах.

Пример: доступ к системе должен осуществляться по HTTPS-протоколу.

Ограничения – требования, которые накладываются на доступные разработчику возможности проектирования или разработки.

Пример: система должна быть реализована на платформе .NET Framework 4.0.

Внешние интерфейсы – требования, которые определяют список интерфейсов в системе, их наполнение и задачи. Требования в этом классе описывают связь вашей системы с остальным миром, касающиеся взаимодействия с пользователями, оборудованием и другими системами ПО.

Пример: сайт должен иметь интеграцию с PayPal для оплаты услуг.

При организации работы с документом необходимо уделить внимание построению структуры документа, используемому формату, сохранности и версионности документа при предоставлении версий документа для использования другими лицами.

Перед началом специфицирования требований следует учитывать, в каких случаях лучше использовать различные представления информации:

- текстовое;
- табличное;
- в виде диаграммы или модели.

В некоторых случаях одного формального документа бывает недостаточно для коммуникации и требуется приготовить краткий обзор в виде презентации. При выборе представления следует иметь в виду, что существуют следующие моделируемые сущности:

- пользовательские классы и роли;
- сущности и связи;
- события;

- процессы;
- правила.

Пользовательские, функциональные и нефункциональные требования обычно включаются в спецификацию требований к программному обеспечению, с которой работают другие заинтересованные лица.

Практическое задание:

1 Разбить сформулированные в лабораторной работе №1 требования на три уровня: бизнес-требования, требования пользователей, функциональные требования.

2 Используя шаблон спецификации требований, оформить все требования вашего проекта.

3 Оформить отчет и защитить лабораторную работу.

Содержание отчета:

1 Цель работы.

2 Спецификация требований вашего проекта.

3 Выводы по работе.

Контрольные вопросы:

1 Какие существуют уровни требований? Дайте характеристику каждому.

2 Что такое бизнес-требования? Приведите пример.

3 Что такое пользовательские требования? Приведите пример.

4 Что такое функциональные требования? Приведите пример.

5 Что такое нефункциональные требования? Приведите пример.

6 Как осуществляется выбор подхода к документированию?

Лабораторная работа №3
Создание прототипа в инструментальной среде

Цель: изучить виды прототипов и цели их создания, ознакомиться с инструментами для прототипирования, научиться создавать прототип в инструментальной среде в зависимости от контекста задачи и иных условий.

План занятия:

1 Изучить теоретические сведения.

2 Выполнить практическое задание по лабораторной работе.

3 Оформить отчет и ответить на контрольные вопросы.

Теоретические сведения

Прототип – это модель внешнего вида или симуляция поведения программного продукта. Это частичное, возможное и предварительное воплощение предлагаемого нового продукта.

Создание прототипа снижает вероятность неправильного восприятия программного продукта, при грамотном подходе может экономить время и усилия на создание программного продукта, создает быструю цепь обратной связи, снижая риски.

Прототип позволяет заинтересованным в проекте лицам исследовать различные варианты реализации взаимодействия пользователей, наглядно представить конечный продукт, оптимизировать удобство работы и оценить возможные технические приемы, помогает обнаружить ошибки и оценить точность и качество требований.

Зачем нужен прототип? Прототип помогает общению с командой и/или заказчиком, наглядно демонстрирует структуру, поведение и дизайн продукта, используется для пользовательского тестирования, документации структуры, поведения и дизайна, согласования структуры, поведения и дизайна.

Исходя из целей создания, прототипы можно разделить на следующие виды:

- высокодетализированные;
- слабодетализированные;
- статичные;
- динамичные (интерактивные).

Различают классы атрибутов прототипа:

1 *По назначению:* модель и экспериментальный образец.

Модель:

а) показывает внешний вид экранов пользовательского интерфейса и позволяет осуществлять частичную навигацию между ними, но не содержит никакой или почти никакой реальной функциональности;

б) позволяет пользователям выяснить, смогут ли они с помощью системы выполнять свою работу;

в) дает возможность обнаружить упущения, неверные и ненужные функции.

Экспериментальный образец:

а) воплощает срез функциональности приложения от интерфейса пользователя через все уровни технических сервисов (действует как настоящая система);

б) позволяет уточнить архитектуру, оптимизировать алгоритмы, оценить предлагаемую схему базы данных, проверить критически важные временные требования.

2 По использованию в будущем: одноразовый и эволюционный.

Одноразовый прототип:

а) разрешает неясности и улучшает требования к ПО;

б) создается при возникновении сложностей с наглядным представлением системы на основе одних требований; помогает решить, годятся ли требования для создания продукта (может выявить пробелы в документации).

Эволюционный прототип:

а) следует создавать для приложений, которые со временем будут расширяться (нет реализации всей запланированной функциональности);

б) уроки, извлеченные из реакции пользователей на тестирование и первоначальное использование продукта, учитываются при модификации в следующем цикле.

3 *По форме:* бумажный и электронный.

Бумажный прототип:

а) позволяет выяснить, как может выглядеть некий фрагмент системы;

б) помогает установить, действительно ли пользователи и разработчики одинаково понимают требования;

в) дает возможность сделать без риска решение продукта до разработки производственного кода продукта.

Электронный прототип:

а) позволяет легко реализовать и обновить компоненты интерфейса пользователя;

б) помогает уточнить требования или пересмотреть решения перед проектированием детализированных интерфейсов;

в) дает возможность команде лучше управлять ожиданиями клиентов.

Способ комбинирования разных видов прототипов представлен на рисунке 3.1.

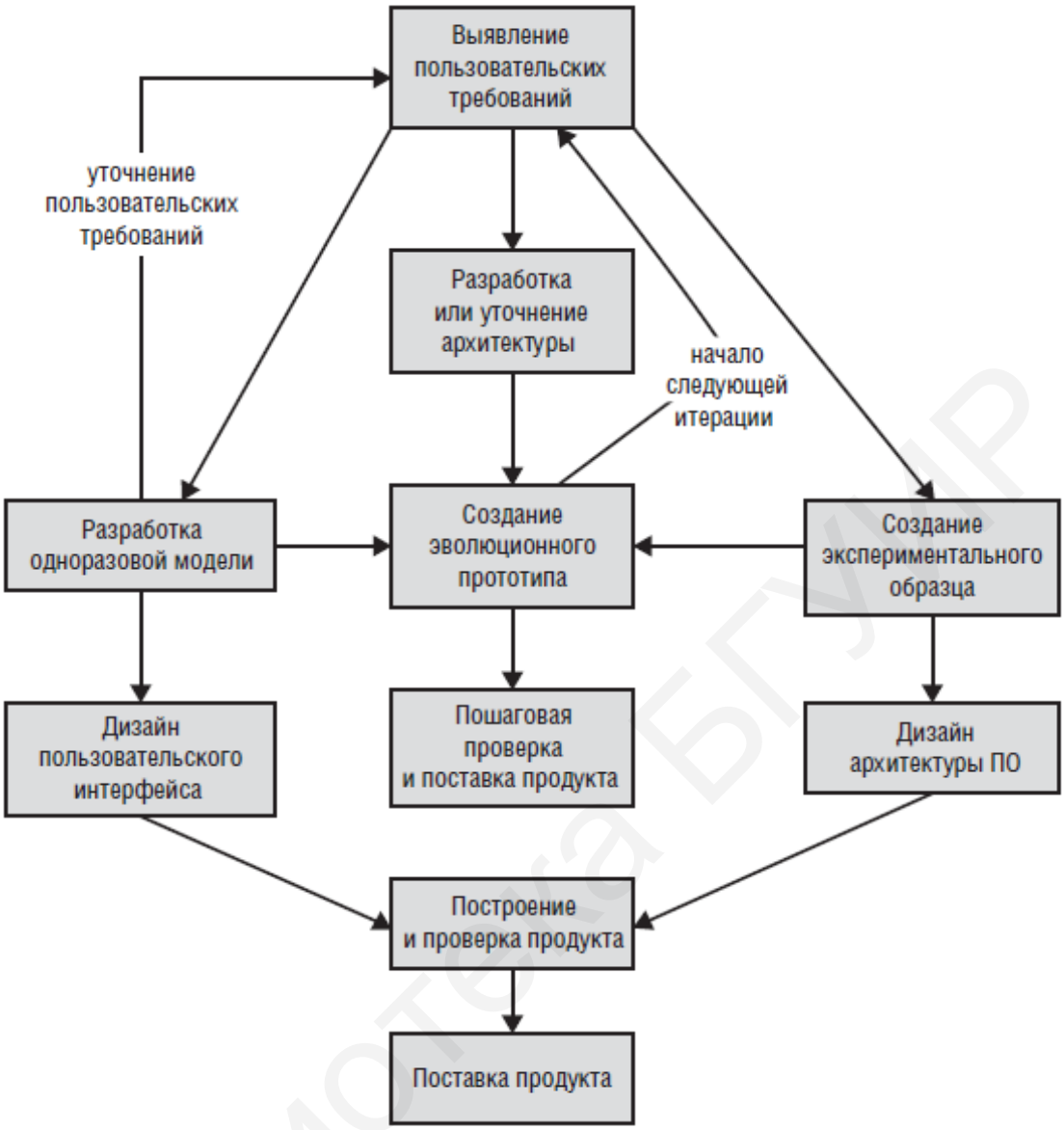


Рисунок 3.1 – Способ комбинирования разных видов прототипов [2]

Прежде чем создавать прототип, нужно принять четкое и ясное решение, прекратится ли работа с прототипом после оценки или он станет частью выпускаемого продукта.

Продуманное применение и грамотная реализация прототипов служит ценным инструментом, помогающим выявлять и уточнять требования в деле преобразования потребностей в решения.

Риски при создании прототипа:

а) подталкивание к выпуску прототипа (нужно управлять ожиданиями, чтобы заинтересованные лица четко понимали цели и ограничения прототипа);

б) отвлечение на детали (нужно ограничиваться только теми экранами, функциями и возможностями навигации, которые помогут устранить неопределенности в требованиях);

в) нереалистичные ожидания производительности (оценка модели не производится в рабочей среде продукта, поэтому производительность конечного продукта отличается от производительности прототипа);

г) слишком много усилий на создание прототипа (нужно относится к прототипу, как к эксперименту: нужна проверка гипотезы; уверенность в том, что требования определены в достаточном объеме, решены архитектурные вопросы и что можно переходить к проектированию и конструированию).

Цель создания прототипа приведена в таблице 3.1. Схемы отражают структуру, прототипы – опыт пользователя, а макеты – внешний вид.

Таблица 3.1 – Цель создания прототипа

Вид	Детализация	Применение
Wireframe (схема)	Низкая	Облегчение понимания, документация
Prototype (прототип)	Средняя или высокая	Показать динамику, тестирование
Mockup (макет)	Средняя или высокая	Чистовой дизайн, согласование

Степень детализации прототипа представлена на рисунке 3.2.

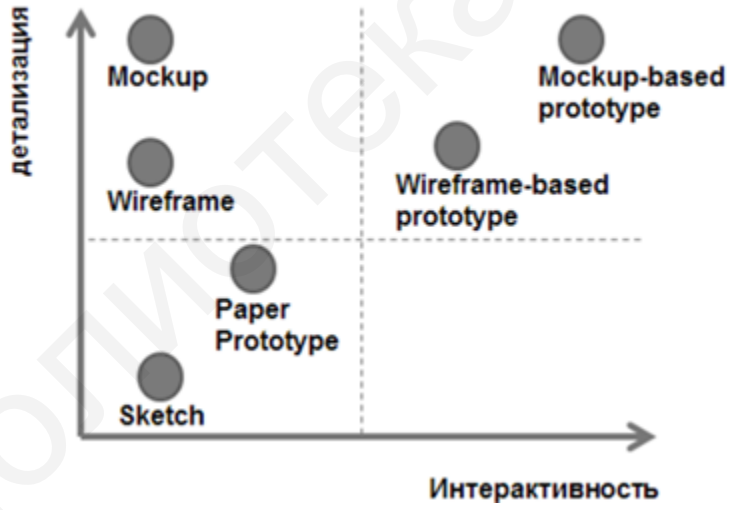


Рисунок 3.2 – Степень детализации прототипа [1]

Последовательность шагов при разработке прототипа представлена на рисунке 3.3.

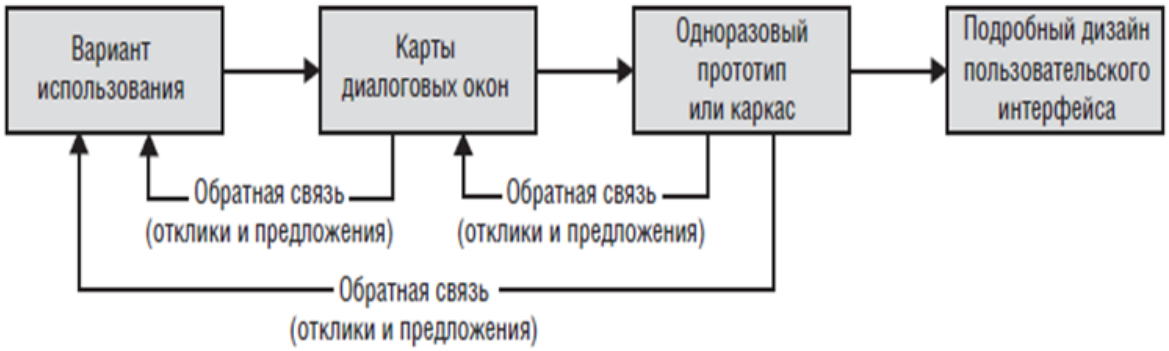


Рисунок 3.3 – Последовательность шагов разработки прототипа

Рассмотрим более подробно последовательность шагов при разработке прототипа:

1 Необходимо сформулировать соответствующие варианты использования (рисунок 3.4).

Класс пользователей	**Вариант использования**
Пользователь	Получить информацию о книге
	Получить информацию об авторе
	Читать избранные книги
	Читать блог
	Связаться с автором
Клиент	Заказать продукт
	Загрузить электронную версию продукта
	Обратиться за помощью в решении проблемы
Администратор	Управлять списком продуктов
	Осуществлять возврат средств клиенту
	Управлять списком сообщений электронной почты

Рисунок 3.4 – Пример варианта использования

2 Необходимо создать карту диалоговых окон, обдумать, какие страницы должны быть и представить пути навигации между ними (рисунок 3.5).

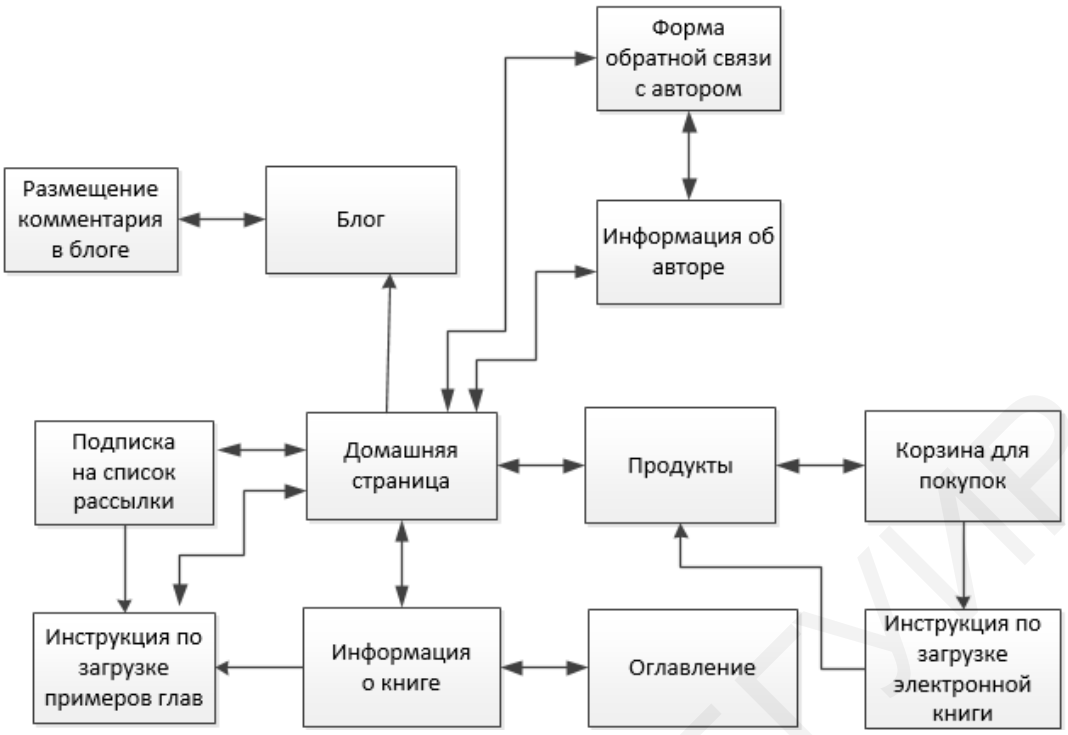


Рисунок 3.5 – Пример карты диалоговых окон

3 Следует построить одноразовый прототип или каркас избранных страниц для выработки концепции визуального дизайна.

4 Необходимо создать подробный дизайн пользовательского интерфейса.

Рекомендации по созданию прототипов:

а) включайте задачи создания прототипов в план своего проекта. Составьте график распределения затрат времени и средств на разработку, оценку и модификацию прототипов;

б) не создавайте прототипы элементов, которые уже понимаете, кроме случаев исследования альтернативных вариантов дизайна;

в) планируйте создание нескольких прототипов. В большинстве случаев не удастся создать то, что нужно, с первой попытки;

г) не ожидайте, что прототип полностью заменит спецификацию требований к ПО.

Распространенные инструменты создания прототипов:

- карандаш и бумага;
- MS Visio;
- Axure RP;
- Balsamiq Mockups;
- Flairbuilder;
- Moqups;
- Draw.io.

Практическое задание:
1 Ознакомиться с инструментом для прототипирования Axure.
2 Создать прототип вашего проекта.
3 Оформить отчет и защитить лабораторную работу.

Содержание отчета:
1 Цель работы.
2 Скриншоты пользовательского интерфейса вашего проекта.
3 Выводы по работе.

Контрольные вопросы:
1 Что такое прототип?
2 Зачем нужен прототип?
3 Что такое модель?
4 Что такое экспериментальный образец?
5 Что такое эволюционный прототип?
6 Какие проблемы могут возникнуть при создании прототипа?

Лабораторная работа №4
Моделирование IT-проекта

Цель: изучить суть визуального моделирования и его применимость на различных этапах процесса разработки программного обеспечения, освоить список наиболее популярных нотаций моделирования в сфере разработки ПО, научиться составлять диаграмму BPMN.

План занятия:

1 Изучить теоретические сведения.

2 Выполнить практическое задание по лабораторной работе.

3 Оформить отчет и ответить на контрольные вопросы.

Теоретические сведения

Модель – упрощенное представление о реальном объекте или процессе.

Информационная модель – совокупность информации, характеризующая свойства и состояния объекта, процесса, явления, а также взаимосвязь с внешним миром.

Моделируемые сущности (таблица 4.1):

- пользовательские классы и роли;
- сущности и связи;
- события;
- процессы;
- правила.

Таблица 4.1 – Описание моделируемых сущностей

Моделируемые сущности	Отображение в моделях
Пользовательские классы и роли	Изначально выявляются при проведении анализа заинтересованных лиц и описываются в организационной модели, процессах/процедурах и вариантах использования
Сущности и связи	Модель данных
События	События служат основой для модели, описывающей рамки решения, но также отражаются в процессах/процедурах, диаграммах состояний и вариантах использования
Процессы	Отражаются в процессах/процедурах, организационных моделях, диаграммах состояний и вариантах использования
Правила	Обычно описываются как бизнес-правила, хотя могут быть включены в процессы/ процедуры, диаграммы состояний и варианты использования

Нотация – система условных обозначений, принятая в какой-либо области знаний или деятельности, включающая множество символов, используемых для представления понятий и их взаимоотношений и составляющих алфавит нотации, а также правила их применения.

Под методологией (нотацией) создания модели (описания) понимается совокупность способов, при помощи которых объекты реального мира и связи между ними представляются в виде модели. Для каждого объекта и связей характерны ряд параметров, или атрибутов, отражающих определенные характеристики реального объекта.

В настоящее время существует множество различных нотаций для описания моделей. Системы обозначения, представленные здесь, обеспечивают общий, стандартный для всей индустрии язык, который используют участники проектов. Не запрещено придумывать свою UML, BPMN, чтобы дополнить возможности письменного общения в рамках проекта, однако не факт, что пользователи интерпретируют их одинаково.

Способы описания моделей:

1) словесные (текстовое описание);

2) табличные (текст, структурированный в виде матрицы или таблицы);

3) графические модели визуального представления.

Выделяют следующие виды графического описания моделей:

1 Общие:

а) диаграммы потоков данных (data flow diagrams, DFD);

б) модели данных:

- диаграммы «сущность – связь» (entity-relationship diagrams, ERD);
- диаграммы классов;

в) диаграммы перехода состояний (state-transition diagrams, STD), называемые также диаграммами состояний;

г) диаграммы вариантов использования (use case diagrams);

д) диаграммы взаимодействия (sequence diagrams);

е) карты диалогов (dialog maps) или карты перемещений (navigation map);

ж) диаграммы процессов:

- блок-схемы;
- процедуры (flowcharts);
- диаграммы деятельности (activity diagram).

2 Специализированные:

а) BPMN (Business Process Model and Notation) – описывает функциональную последовательность работ;

б) EPC (Event-Driven Process Chain – событийная цепочка процессов) – описывает событийную последовательность работ;

в) IDEF0 – описывает логическую последовательность работ.

Рассмотрим некоторые из них.

Диаграмма потока данных (data flow diagram, DFD) – методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хра-

нилища данных, к которым осуществляется доступ. Это один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML. Несмотря на имеющее место в современных условиях смещение акцентов от структурного к объектно-ориентированному подходу к анализу и проектированию систем, «старинные» структурные нотации по-прежнему широко и эффективно используются как в бизнес-анализе, так и в анализе информационных систем.

Модель DFD, как и большинство других структурных моделей, – иерархическая модель. Каждый процесс может быть подвергнут декомпозиции, т. е. разбиению на структурные составляющие, отношения между которыми в той же нотации могут быть показаны на отдельной диаграмме. Когда достигнута требуемая глубина декомпозиции, процесс нижнего уровня сопровождается мини-спецификацией (текстовым описанием). Кроме того, нотация DFD поддерживает понятие подсистемы – структурного компонента разрабатываемой системы.

Области применения нотации DFD:

а) определение существующих хранилищ данных (текстовых документов, файлов, баз данных);

б) определение и анализ данных, необходимых для выполнения каждой функции процесса;

в) подготовка к созданию модели структуры данных организации;

г) выделение основных и вспомогательных процессов организации DFD.

Основные объекты нотации (на примере Yourdon Notation) представлены в таблице 4.2 и на рисунке 4.1.

Таблица 4.2 – Описание объектов нотации DFD

Графический вид	Название	Описание
	Работа (Process)	Отображают функции (работы, процессы)
	Стрелка (Arrow)	Отображают потоки данных между работами
	Хранилище данных/файл (Data Store)	Отображают любой носитель информации (бумажный документ, файл, базу данных на сервере)
	Внешняя сущность (External Entity)	Отображают внешние объекты, с которыми происходит взаимодействие (например, заказчик, поставщик, партнер и т. д.)

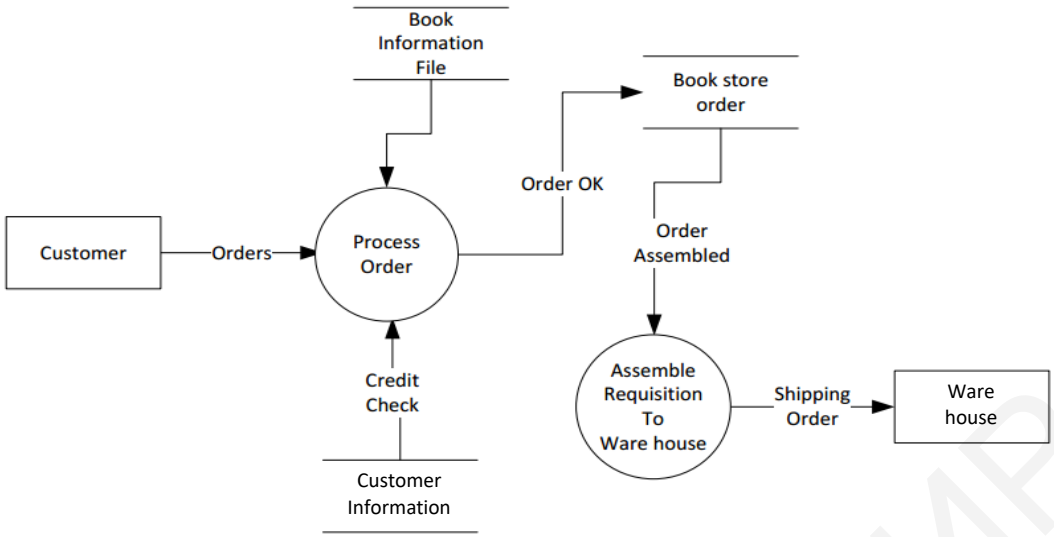


Рисунок 4.1 – Пример диаграммы потока данных

Диаграмма перехода состояний (state-transition diagram, STD) – моделирование поведения системы, зависящего от времени или реакций системы на некоторые события (рисунок 4.2).

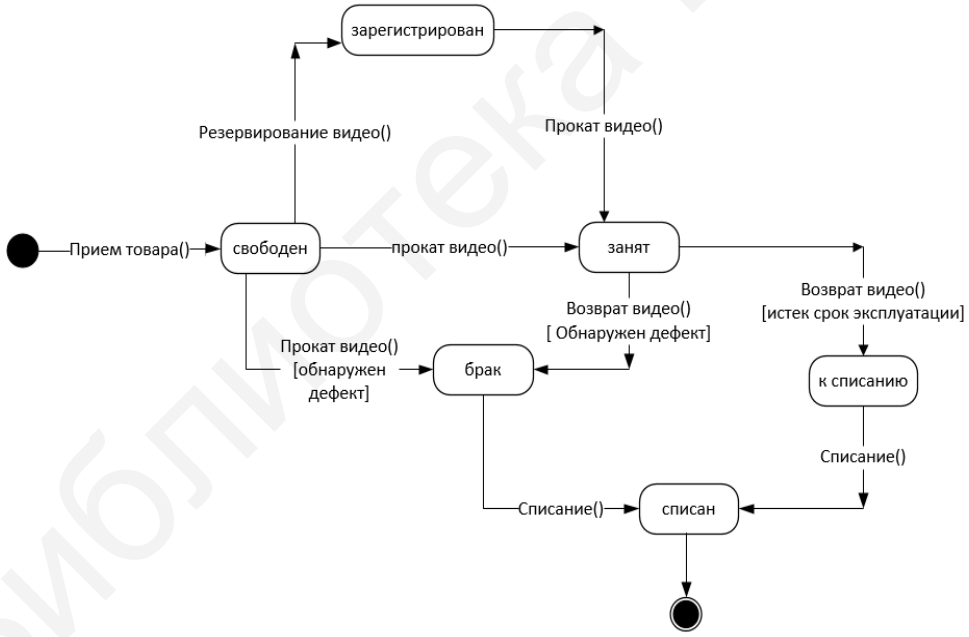


Рисунок 4.2 – Пример диаграммы перехода состояний

STD состоит из следующих объектов:

1 Состояние – моделируемая система в любой заданный момент времени должна находиться точно в одном из конечного множества состояний.

2 Начальное состояние – частный случай состояния, которое не содержит никаких внутренних действий (псевдосостояния). В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания

на диаграмме состояний графической области, от которой начинается процесс изменения состояний. В противном случае переход никак не помечается. Если этот переход не помечен, то он является первым переходом в следующее за ним состояние.

3 Конечное (финальное) состояние – частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояния). В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени. Оно служит для указания на диаграмме состояний графической области, в которой завершается процесс изменения состояний или жизненный цикл данного объекта.

4 Простой переход (simple transition) – отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния другим. Пребывание моделируемого объекта в первом состоянии может сопровождаться выполнением некоторых действий, а переход во второе состояние будет возможен после завершения этих действий, а также после удовлетворения некоторых дополнительных условий.

5 Действие – это операция, которая может быть связана с переходом, и выполняющаяся при выполнении перехода.

На STD состояния представляются узлами, а переходы – дугами. Условия идентифицируются именем перехода и возбуждают выполнение перехода. Действия или отклики на события привязываются к переходам и записываются под соответствующим условием. Начальное состояние на диаграмме должно иметь входной переход, изображаемый потоком из стартового узла.

Применяются два способа построения STD: первый способ заключается в идентификации всех возможных состояний и дальнейшем исследовании всех не бессмысленных связей (переходов) между ними. По второму способу сначала строится начальное состояние, затем следующие за ним и т. д.

В ситуации, когда число состояний и/или переходов велико, для проектирования спецификаций управления могут использоваться матрицы переходов состояний. В матрице переходов по вертикали указываются состояния, из которых осуществляется переход, а по горизонтали – состояния, в которые осуществляется переход. При этом каждый элемент матрицы содержит соответствующие условия и действия, обеспечивающие переход из вертикального состояния в горизонтальное.

Диаграммы процессов:

1) процесс (Basic Flowchart);

2) процедура (Cross Functional Flowchart).

Нотация **процесс (Basic Flowchart)**, известная также как блок-схема, – наверное, самый простой способ графического представления выполнения любого процесса. Блок-схемы часто применяются в учебных целях для отображения алгоритма выполнения какой-либо задачи. Как оказалось, диаграммы в формате Flowchart могут также успешно применяться для построения бизнес-процессов, для этого в схему, как правило, вводят несколько дополнительных элементов: ответственность и ресурсы.

Диаграмма читается сверху вниз, по направлению, которое указывается стрелочками. Дополнительно над каждой линией может писаться комментарий, поясняющий выполнение следующего действия, или объект, который задействован при выполнении данной функции (рисунок 4.3). В нотации Flowchart можно отобразить не только последовательность выполнения функций, но и их цикличность, для чего используется блок в виде ромба с несколькими выходами. В случае если возникает неоднозначность выполнения следующей функции, например, когда из блока выходит сразу несколько стрелочек, принято над линией писать условие, при котором выполняется следующая функция.

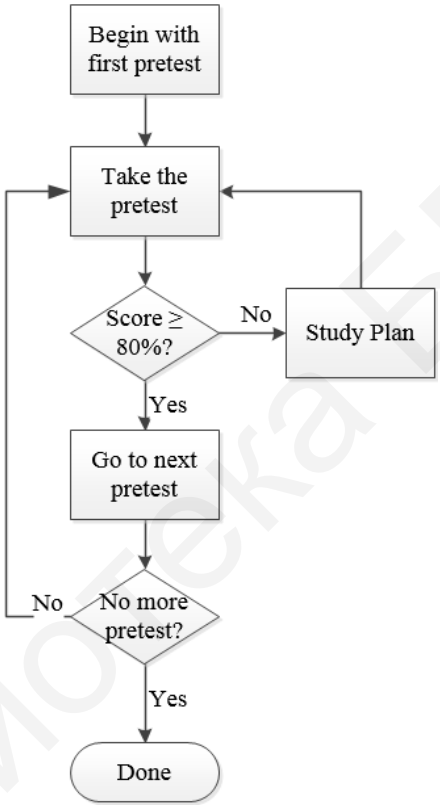


Рисунок 4.3 – Пример Basic Flowchart

Данная нотация используется для представления алгоритма выполнения процесса (нотация класса workflow). Используются графические элементы: событие, процесс, решение, два типа стрелок – стрелки предшествования и стрелки «Поток объектов». Процесс (Basic Flowchart) состоит из прямоугольников (бизнес-процессы), в которые входят и из которых выходят стрелки (потоки информации, документов). Также в нотации используются элементы типа «Решение», которые позволяют делать ветвления. Для обозначения начала выполнения всего бизнес-процесса и его окончания могут быть использованы фигуры типа «Событие» (элементы, похожие на овалы).

Каждый бизнес-процесс в нотации может быть декомпозирован (разбит на детальные бизнес-процессы).

Преимущество Basic Flowchart – простота и наглядность. С ее помощью можно быстро описать шаги бизнес-процесса. Использование не требует специальных знаний, т. к. легко воспринимается сотрудниками с разным уровнем подготовки.

Недостаток Basic Flowchart – простота. Набор графических элементов очень ограничен для передачи информации о бизнес-процессе. Например, на диаграмме никак не обозначены участники бизнес-процесса (это с успехом решено в Cross Functional Flowchart).

Существуют различные варианты реализации нотации Flowchart в современных системах для бизнес-моделирования. Один из таких вариантов – Cross Functional Flowchart.

Нотация **процедура (Cross Functional Flowchart, функциональная блок-схема, кросс-функциональная схема)** служит для отображения процесса на нижнем уровне бизнес-модели. Из-за своей простоты и удобства она является одной из самых используемых нотаций среди пользователей.

Cross Functional Flowchart отображает детальный алгоритм выполнения бизнес-процесса и всех его участников, а также способы их взаимодействия между собой в рамках процедуры. Дорожка на диаграмме отражает должность исполнителя, подразделение и роль в процессе. На дорожках Cross Functional Flowchart размещаются действия, за которые и отвечает исполнитель. Каждое действие может быть декомпозировано (разбито на более детальные бизнес-процессы).

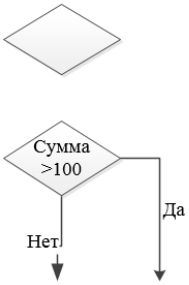
Действия на дорожках процедуры связаны между собой информационными или материальными потоками. Дорожки на кросс-функциональной схеме могут быть как горизонтальные, так и вертикальные. Выбор направления зависит от стандарта предприятия или вкуса разработчика Cross Functional Flowchart. В процедуре так же могут использоваться решения (условия) для ветвления бизнес-процесса.

Преимущество Cross Functional Flowchart – простота создания и понимания сотрудниками с разным уровнем подготовки.

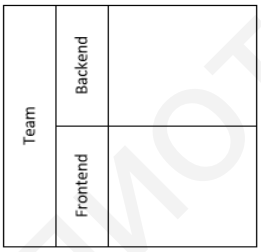
Недостатки Cross Functional Flowchart: в некоторых случаях может быть удобней использовать дополнительные графические элементы, которые имеются в других нотациях. Также недостатком такой диаграммы можно считать неудобство отображения большого количества участвующих в процессе должностей.

Используемые в Cross Functional Flowchart графические символы представлены в таблице 4.3, пример – на рисунке 4.4.

Таблица 4.3 – Описание объектов нотации Cross Functional Flowchart

Название	Графический символ	Описание
1	2	3
Действие		Действие обозначается с помощью прямоугольного блока. Внутри блока помещается название действия. Временная последовательность выполнения действий задается расположением действий на диаграмме процесса/процедуры сверху вниз (слева направо на горизонтальной диаграмме Cross Functional Flowchart)
Логический блок (блок условия)		Отображает решение или функцию переключательного типа с одним входом и двумя или более альтернативными выходами, из которых только один может быть выбран после вычисления условий, определенных внутри этого элемента. Вход в элемент обозначается линией, входящей обычно в верхнюю вершину элемента. Если выходов два или три, то обычно каждый выход обозначается линией, выходящей из остальных вершин (боковых и нижней). Если выходов больше трех, то их следует показывать одной линией, выходящей из вершины (чаще нижней) элемента, которая затем разветвляется. Соответствующие результаты вычислений могут записываться рядом с линиями, отображающими эти пути
Событие		События отображают стартовые точки процесса/процедуры, приводящие к началу выполнения действия/процедуры, и конечной точки, наступлением которой заканчивается выполнение процесса/процедуры. Началом процесса/процедуры считается событие, из которого только исходят стрелки передачи управления. Концом процесса/процедуры считается событие, в которое только входят стрелки передачи управления
Данные (вход-выход)		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод). Данный символ не определяет носителя данных, для указания типа носителя данных используются специальные символы

Продолжение таблицы 4.3

1	2	3
Связь предшествования		Стрелки «Связь предшествования» обозначают передачу управления от одного действия к другому, т. е. предыдущее действие должно закончиться прежде, чем начнется следующее. Стрелка, запускающая выполнение действия, изображается входящей в действие сверху. Стрелка, обозначающая передачу управления другому (другим) действию, изображается выходящей из действия снизу. Если стрелка служит только для обозначения передачи управления, то имя стрелки оставляется пустым. Если кроме передачи управления из предыдущего действия в следующее действие поступает Объект(ы), то стрелка именуется и в список объектов стрелки заносится соответствующий Объект(ы)
Дорожка (диаграмма Cross Functional Flowchart)		Дорожки предназначены для отображения организационных единиц (должности, подразделения, роли) – исполнителей действий процедуры
Предопределенный процесс		Символ отображает выполнение процесса, состоящего из одного или нескольких операций, который определен в другом месте программы (в подпрограмме, модуле). Внутри символа записывается название процесса и передаваемые в него данные

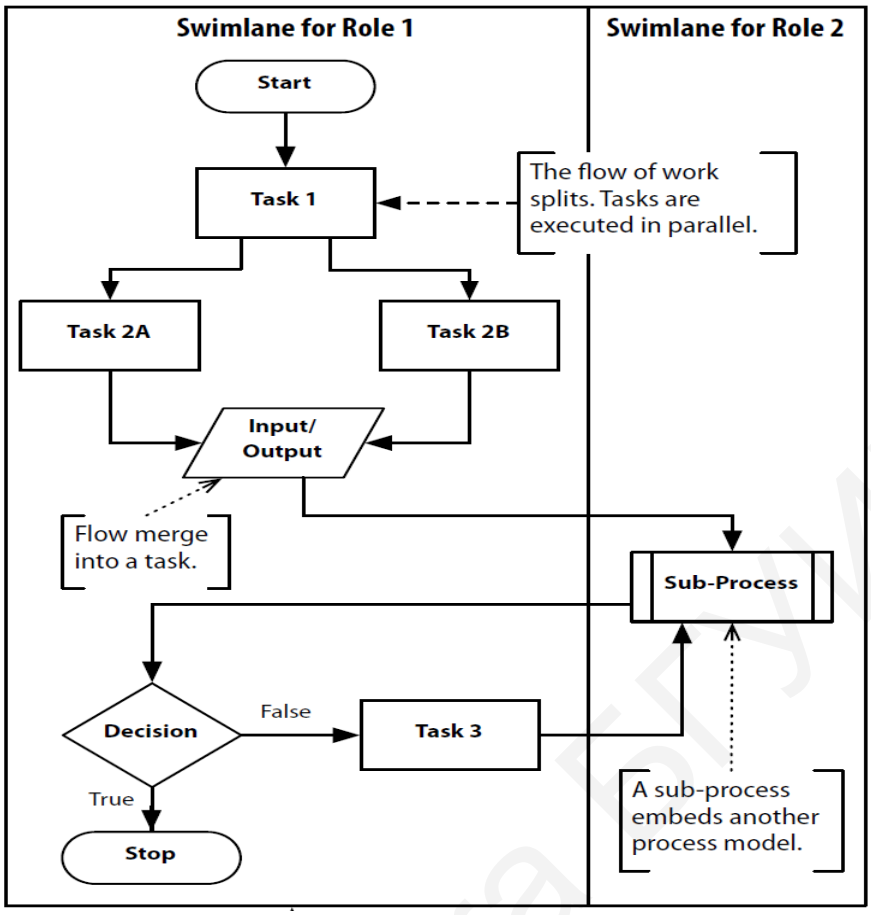
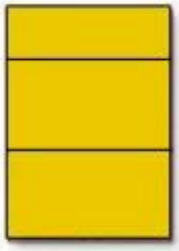


Рисунок 4.4 – Пример Cross Functional Flowchart [1]

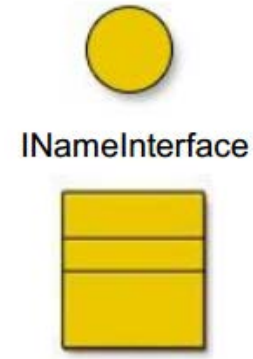
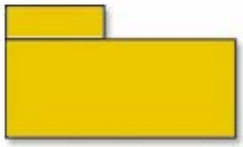
Нотация UML

UML – язык графического описания для объектного моделирования в области разработки программного обеспечения. UML описывает модель реального мира и будущей системы с помощью различных диаграмм. У каждой диаграммы есть свое назначение и особые объекты (таблица 4.4).

Таблица 4.4 – Описание объектов нотации UML

Название	Графический символ	Описание
1	2	3
Классы (Class)		Включает набор объектов с одинаковыми атрибутами, операциями, отношениями и семантикой. Класс реализует один или несколько интерфейсов и изображается в виде прямоугольника, включающего имя класса, атрибуты, операции

Продолжение таблицы 4.4

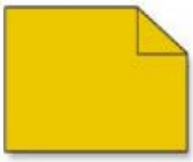
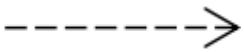
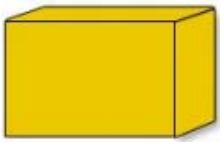
1	2	3
Интерфейсы (Interface)		Это именованное множество операций, которые характеризуют поведение отдельного элемента модели. Изображается окружностью. Для представления внутренней структуры интерфейса используют прямоугольник класса. Имя интерфейса рекомендуется записывать на английском языке и начинать с заглавной буквы
Прецеденты (Use case)		Описывают набор последовательностей действий, которые выполняются системой и имеют значение для конкретного действующего лица (Actor)
Пакеты (Packages)		Основной способ организации элементов модели в языке UML. Включает в себя однотипные элементы (классы, интерфейсы и т. д.) или пакеты

Рассмотрим основные элементы для диаграмм нотации UML (таблица 4.5).

Таблица 4.5 – Описание основных элементов для диаграмм нотации UML

Название	Графический символ	Описание
1	2	3
Состояния (State)		Состояние – это модель отдельной ситуации, в течение которой имеет место выполнение некоторого условия. Включает имя состояния и список внутренних действий в данном состоянии
		Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий. В этом состоянии находится объект по умолчанию в начальный момент времени
		Конечное (финальное) состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояния). В этом состоянии будет находиться объект по умолчанию после завершения всех возможных состояний

Продолжение таблицы 4.5

1	2	3
Компоненты (Component)		Компонент – это физическое представление таких логических элементов, как классы, интерфейсы и кооперации. Предметная область компонентов относится к реализации. Как правило, имеют только имя
Примечания (Note)		Отображает дополнительную информацию: заметки, комментарии и т. п.
Отношения		Ассоциация (Association) – структурное отношение, описывающее множество связей между объектами классификаторов, где связь (Link) – это соединение между объектами, которое описывает связи между их экземплярами. Ассоциации являются как бы клеем, который связывает систему воедино. Без ассоциаций мы имели бы просто некоторое количество классов, не способных взаимодействовать друг с другом
		Зависимость (Dependency) – это семантическое отношение между двумя сущностями, при котором изменение одной из них (независимой сущности) может отразиться на семантике другой (зависимой)
		Обобщение (Generalization) – это отношение специализации/обобщения, при котором объекты специализированного элемента (потомка – Child) можно подставить вместо объектов обобщенного элемента (родителя, предка – Parent). В случае обобщения классов прямой предок может именоваться суперклассом, а прямой потомок – подклассом
Узлы (Block)		Физические объекты, которые существуют во время исполнения программы и представляют собой коммуникационный ресурс, обладающий по крайней мере памятью, а зачастую и процессором. На узлах могут находиться выполняемые объекты и компоненты. Изображаются узлы в виде куба, имеют имя и примечание

Нотация BPMN

Основными элементами BPMN являются:

- элементы потока (события, действия/процессы и шлюзы) (таблицы 4.6–4.9);
- данные (объекты данных и базы данных) (таблица 4.10);
- соединяющие элементы (потоки управления, потоки сообщений и ассоциации) (таблица 4.11);
- зоны ответственности (пулы и дорожки) (таблица 4.12).

Таблица 4.6 – Описание элементов потока

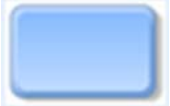
Название	Графический символ	Описание
События (Events)		Отображают начало процесса
		Отображают промежуточное событие
		Отображают завершение процесса
Действия (Activities)		Отображают процесс (Process), подпроцесс (Sub-Process) или задачу (Task)
Шлюзы (Gateways)		Отображают схождения и расхождения потока задач

Таблица 4.7 – Описание элементов событий

События	Начало			Промежуточный этап				Конец
	Стандарт	Прерывающий событийный подпроцесс	Непрерывающий событийный подпроцесс	Обрабатывающие	Граничные прерывающие	Граничные непрерывающие	Генерирующие	Стандарт
1	2	3	4	5	6	7	8	9
Простое: нетипизированное событие, обычно показывающее начало или окончание процесса		–	–	–	–	–		
Сообщение: получение и отправка сообщений								

Продолжение таблицы 4.7

1	2	3	4	5	6	7	8	9
Таймер: циклические временные события, моменты времени, временные периоды и таймеры							–	–
Эскалация: повышение к более высокому уровню организации иерархии	–			–				
Условный: реакция на меняющиеся бизнес-условия или интеграция бизнес-правил							–	–
Ссылка: пара соответствующих ссылок эквивалентна потоку последовательности	–	–	–		–	–		–
Ошибка: генерация и обработка заданного типа ошибок	–		–	–		–	–	
Отмена: обработка отмены транзакции или имитирование отмены	–	–	–	–		–	–	
Компенсация: обработка или инициирование компенсации	–		–	–		–		
Сигнал: передается между процессами и может обрабатываться многими получателями								
Составное: обработка одного события из множества или генерация всех определенных событий								

Продолжение таблицы 4.7

1	2	3	4	5	6	7	8	9
Параллельное составное: обработка всех множеств параллельных событий							–	–
Прекращение: немедленное прекращение выполнения процесса	–	–	–	–	–	–	–	

Таблица 4.8 – Описание элементов действий

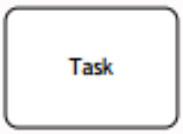
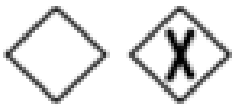
Графический символ	Описание
	Задача – это единица работы, выполняемая работа
	Транзакция – это совокупность логических действий, которые логически принадлежат друг другу
	Событийный подпроцесс, который помещается внутри другого процесса. Он активируется, если инициируется его начальное событие
	Вызывающее действие является точкой входа для глобально определенного подпроцесса, который повторно используется в данном процессе

Таблица 4.9 – Описание элементов шлюзов

Название	Графический символ	Описание
1	2	3
Шлюз исключающего ИЛИ, управляемый данными		При ветвлении направляет поток лишь по одной из исходящих ветвей. При синхронизации потоков оператор ожидает завершения одной входящей ветки и активизирует исходящий поток управления
Шлюз исключающего ИЛИ, событийный		Предшествует только событиям обработки или заданиям обработчикам сообщений. Поток управления направляется по той ветке, где событие произошло раньше

Продолжение таблицы 4.9

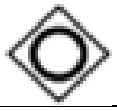
1	2	3
Шлюз И		При разделении на параллельные потоки все ветки активизируются одновременно. При синхронизации параллельных ветвей оператор ждет завершения всех сходящих ветвей и затем активизирует исходящий поток
Шлюз ИЛИ		При ветвлении активируется одна или более ветвей. При слиянии должны быть завершены все выполняющиеся входящие ветви
Шлюз исключающего ИЛИ, событийный (создает новый экземпляр)		Наступление каждого из последующих событий создает экземпляр процесса
Сложный шлюз		Моделирует сложные условия ветвления и слияния
Шлюз И, событийный (создает новый экземпляр)		Наступление всех последующих событий создает экземпляр процесса

Таблица 4.10 – Описание элементов данных

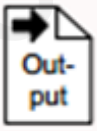
Название	Графический символ	Описание
Объекты данных		Отображает артефакты, результаты выполнения задач. Может быть использован для демонстрации входов/выходов процесса
Хранилище данных		Отображает хранилища информации (картотеки, каталоги, файл, базу данных). Он сохраняется за пределами срока действия экземпляра процесса
Входные данные		Внешний вход для всего процесса. Действия могут использовать эти данные
Выходные данные		Результат всего процесса
Коллекция объектов данных		Группа объектов, несущих информацию, которая отображается в ходе процесса, например, документ или письмо

Таблица 4.11 – Описание соединяющих элементов

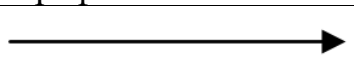
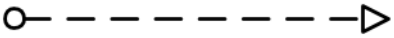
Название	Графический символ	Описание
Соединяющие элементы		Отображают потоки операций
		Отображают связи между информацией и элементами потока
		Отображают обмен сообщениями между пулами
Ассоциация		Используется для отображения связи объектов данных и баз данных с процессами

Таблица 4.12 – Описание элементов зон ответственности

Название	Графический символ	Описание
Аннотация		Отображает дополнительную информацию: заметки, комментарии и т. п.
Пул		Отображает зону всего процесса. Также используется для отображения внешней по отношению к процессу сущности или сервиса
Дорожка		Разделяет пул для группировки задач по зонам ответственности (подразделения или конкретного исполнителя)

Пример описания бизнес-процесса в нотации BPMN представлен на рисунке 4.5

Любая из моделей системы должна содержать только те элементы, которые определены в нотации. Для единого понимания всеми участниками разработки диаграмм используйте только те конструкции, которые уже определены. Не допускается какое бы то ни было переопределение семантики тех элементов, которые отнесены к базовой нотации.

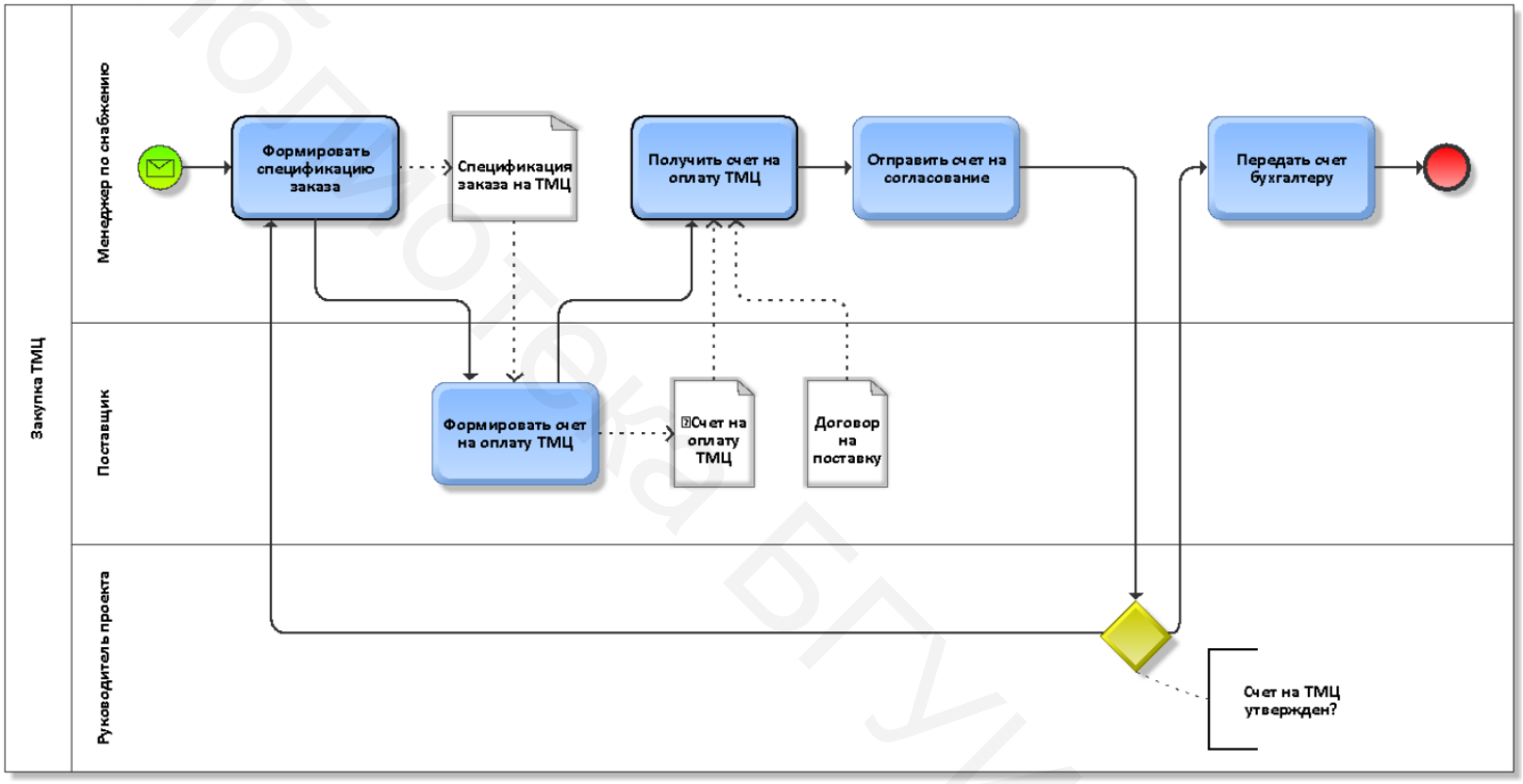


Рисунок 4.5 – Пример описания бизнес-процесса в нотации BPMN [1]

При графическом изображении диаграмм следует придерживаться следующих основных рекомендаций:

1 Каждая диаграмма должна служить законченным представлением соответствующего фрагмента моделируемой предметной области. Отсутствие тех или иных элементов на диаграмме (согласно нотификации) служит признаком неполноты модели и может потребовать ее последующей доработки.

2 Все сущности на диаграмме модели должны быть одного концептуального уровня. Придерживайтесь стратегии последовательного уточнения или детализации отдельных диаграмм; объединяйте диаграммы в пакеты.

3 Вся информация о сущностях должна быть явно представлена на диаграммах. В нотации могут быть использованы значения по умолчанию (например, атрибуты и операции классов), но для однозначного понимания диаграмм всеми участниками разработки, необходимо стремиться к явному указанию свойств всех элементов диаграмм.

4 Диаграммы не должны содержать противоречивой информации. Противоречивость модели может служить причиной серьезнейших проблем при ее реализации и последующем использовании на практике. К противоречиям относятся: замкнутые пути в отношениях агрегирования или композиции; наличие элементов с одинаковыми именами и различными атрибутами свойств; наличие идентичных элементов с разными именами; несогласованность отношений, операций и атрибутов в связанных элементах.

5 Диаграммы не следует перегружать текстовой информацией. Наличие больших фрагментов развернутого текста служит признаком недостаточной проработанности модели или ее неоднородности.

6 Диаграмма должна быть однородна. Типичным нарушением однородности является стремление в рамках одной модели представить различную по характеру информацию.

7 Каждая диаграмма должна быть самодостаточной для правильной интерпретации всех ее элементов и понимания семантики всех используемых графических символов. Любые пояснительные тексты, которые не являются собственными элементами диаграммы (например, комментариями), не должны приниматься во внимание разработчиками. Отдельно расположенные элементы, не имеющие связи с другими элементами системы на диаграмме, – сигнал о неоднородности элементов.

8 Количество диаграмм должно быть достаточным, но не избыточным. Для простых приложений нет необходимости строить все без исключения типы диаграмм. Например, модель системы может не содержать диаграмму развертывания для приложения, выполняемого локально на компьютере пользователя. Важно понимать, что перечень диаграмм зависит от специфики конкретного проекта системы.

Практическое задание:

1 Получить у преподавателя задание для моделирования.
2 Выполнить моделирование вашего проекта.
3 Оформить отчет и защитить лабораторную работу.

Содержание отчета:

1 Цель работы.
2 Нотация процесса вашего проекта.
3 Выводы по работе.

Контрольные вопросы:

1 Что такое модель?
2 Что такое информационная модель?
3 Назовите моделируемые сущности и охарактеризуйте их.
4 Что такое нотация?
5 Назовите способы описания моделей.

Лабораторная работа №5
Анализ и управление проектом

Цель: изучить виды планов и рисков, процессы управления качеством проекта, научиться составлять структуру разбиения работ, диаграмму Ганта и таблицу возможных рисков.

План занятия:

1 Изучить теоретические сведения.

2 Выполнить практическое задание по лабораторной работе.

3 Оформить отчет и ответить на контрольные вопросы.

Теоретические сведения

План – документ, разработанный в соответствии с определенными схемами и методами, описывающий, как, когда и кем будут достигнуты установленные цели.

Виды планов:

1) концептуальный план;
2) стратегический план реализации;
3) тактический (детальный, оперативный) план.

Планирование проекта – непрерывный процесс определения наилучшего способа действий для достижения поставленных целей проекта с учетом складывающейся обстановки.

Планирование позволяет построить модели реализации проекта, координировать деятельность участников проекта, определять порядок, в котором должны выполняться работы.

Процессы планирования – процессы, осуществляемые для тщательного определения содержания проекта, разработки плана управления проектом, идентификации и составления расписания операций проекта, которые будут проводиться в рамках проекта (рисунок 5.1).

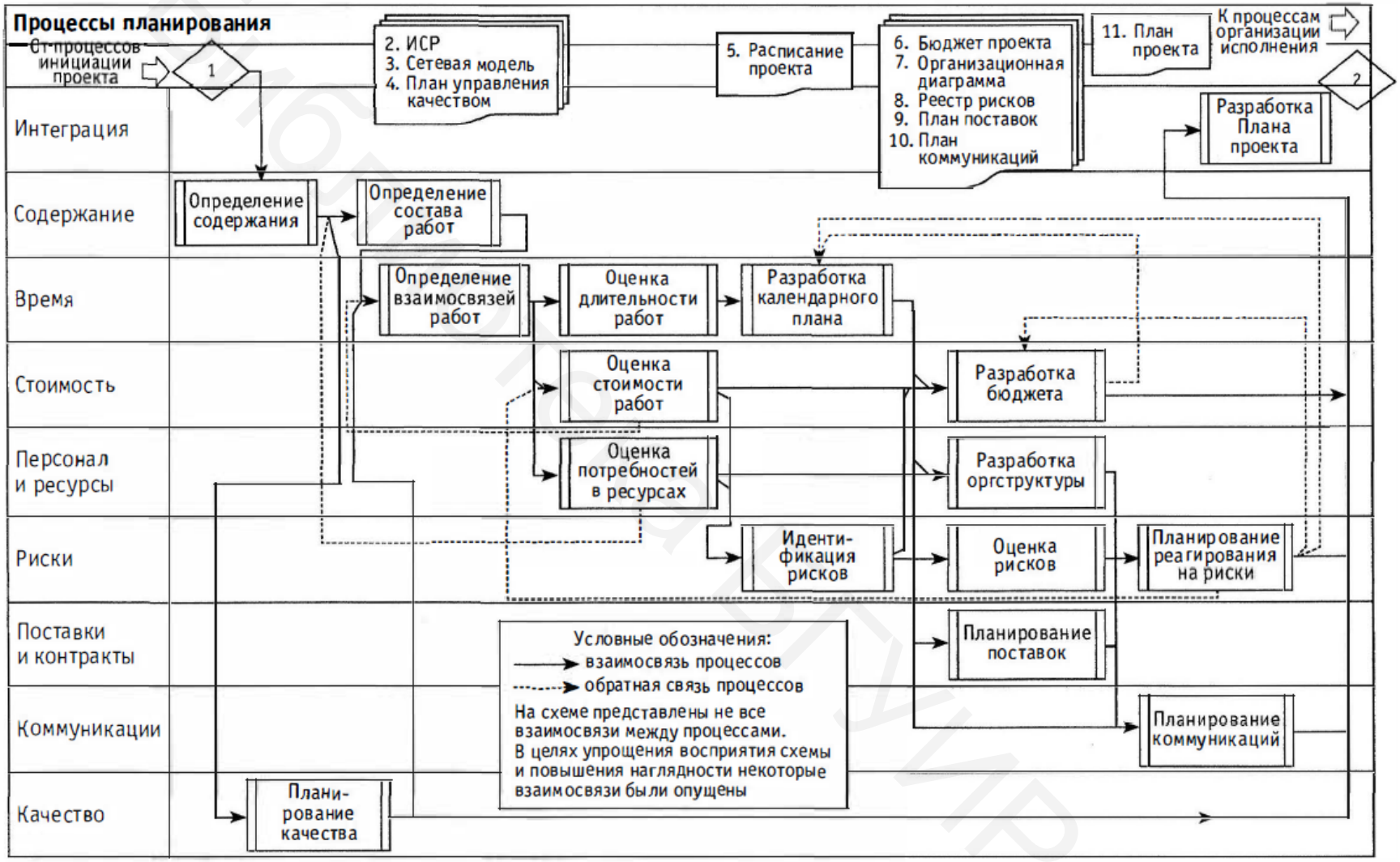


Рисунок 5.1 – Процессы планирования проекта

Структура разбиения работ (Work Breakdown Structure, WBS) – иерархическая структура последовательной декомпозиции проекта на подпроекты, пакеты работ различного уровня, пакеты детальных работ (рисунок 5.2).

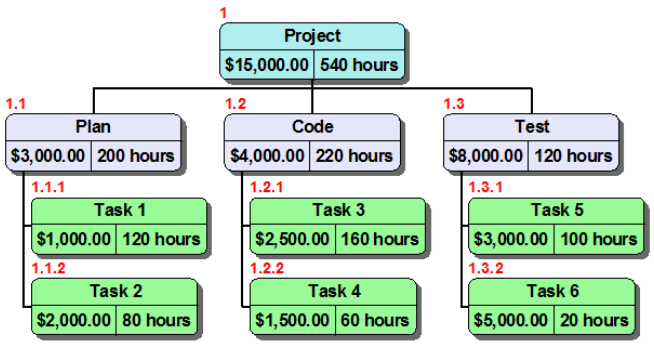


Рисунок 5.2 – Пример иерархической структуры работ

Структура разбиения работ решает проблемы организации работ:

а) распределяет ответственность за достижение целей проекта между его исполнителями;

б) оценивает стоимость;

в) создает систему отчетности, соответствующую целям проекта;

г) поддерживает процедуру сбора информации о выполнении работ;

д) отображает результаты в информационной системе.

Структура разбиения работ должна стать рабочим инструментом как для управления и согласования позиций на совещаниях, так и для сдачи работ, особенно когда сроки были сорваны, а бюджет превышен по независящим от команды проекта причинам (таблица 5.1, рисунок 5.3).

Уровень детализации зависит от сложности и размеров проекта.

Таблица 5.1 – Примерный план разбиения работ

Операции	Предшествующие операции	Длительность
A. Установка компьютеров	–	1
B. Протяжка сети	–	2
C. Настройка сети	A, B	3
D. Установка ПО	C	1
E. Разработка регламента использования ПО	–	4
F. Обучение пользователя	D, E	3

Метод критического пути – расчет возможного календарного графика выполнения комплекса работ на основе описанной логической структуры сети и оценок продолжительности выполнения каждой работы.

Критический путь – максимальный по продолжительности полный путь в сети.

Диаграмма Ганта – это горизонтальная линейная диаграмма, на которой задачи проекта представляются протяженными во времени отрезками, характеризующимися датами начала и окончания, задержками и, возможно, другими временными параметрами.

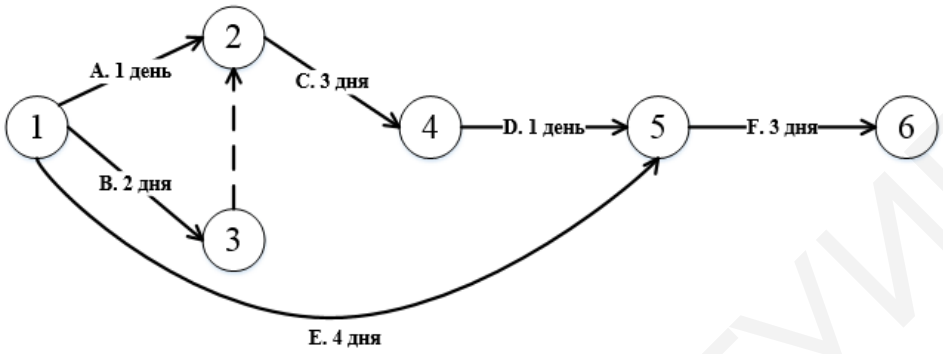


Рисунок 5.3 – Пример построения критического пути процесса

Риск – это условие со степенью неопределенности, которое может повлечь какие-либо потери или другим способом поставить под угрозу успех проекта.

Виды рисков:

1 Бизнес-риски:

а) финансовые риски:

- риск увеличения стоимости проекта;
- риск неполной или несвоевременной оплаты стоимости проекта;

б) информационные риски:

- утечки конфиденциальной информации;
- неполная информация о проекте или бизнесе;
- несвоевременное предоставление информации;

в) маркетинговые риски:

- ошибка маркетинговых расчетов, целей;
- перенос сроков выхода на рынок;
- ошибка в оценке конкурентов;

г) риски бизнес-процессов:

- ошибки в самих процессах (неадекватность структуры бизнеса);
- дисбаланс полномочий и ответственности.

2 Технологические риски:

а) особенности и ограничения технологий;

б) сложности качественно новых технологий.

3 Риски, связанные с требованиями:

а) выявление требований (отсутствие V&S, не определены нефункциональные требования, решения, предлагаемые в качестве потребностей);

б) анализ требований (отсутствие приоритетов, сложные решения);

в) спецификация требований (неоднозначная терминология, отсутствие единого понимания требований);

г) утверждение требований (несогласованные требования, отсутствие проверки);

д) управление требованиями (изменение требований, увеличение объема проекта).

4 Риски, связанные с квалификацией:

а) квалификация команды разработчиков;

б) квалификация менеджера проекта;

в) опыт ведения аналогичных проектов;

г) квалификация сотрудников заказчика.

Оценка рисков:

1) *условие, причина*. Содержит описание существующего фактора или особенности проекта, которые могут спровоцировать нежелательную для проекта ситуацию;

2) *последствие*. Описывает ту нежелательную ситуацию, которую следует избежать;

3) *воздействие.* Показывает общий уровень опасности риска, вбирая в одну числовую величину информацию о возможности реализации риска и уровне его угрозы.

Одним из простых инструментов для оценки вероятности возникновения риска и силы воздействия, если риск сработает, является матрица 2×2. Каждый риск располагается вдоль соответствующих осей и таким образом попадает в один из четырех квадратов. Стратегия работы с рисками представлена на рисунке 5.4.

Очевидно, что риски с низкой вероятностью срабатывания и слабым воздействием на проект в случае их срабатывания в большинстве случаев вообще не стоит рассматривать и тратить на них силы и время. В то же время рискам с высокой вероятностью срабатывания и сильным воздействием на проект в случае срабатывания нужно уделить приоритетное и пристальное внимание.

Рисунок 5.4 – Работа с рисками

Существует несколько основных стратегий работы с негативными рисками:

1) избежать риска (Avoid) – принять превентивные меры, чтобы максимально снизить вероятность наступления риска и избежать его последствий;

2) снизить последствия в случае срабатывания риска (Mitigate) – допустить, что риск может сработать, однако продумать и принять меры по минимизации его последствий;

3) передать его третьей стороне (Transfer). Примером такой стратегии может быть заказная разработка или тестирование ПО;

4) принять последствия риска в случае его возникновения (Accept) – эта стратегия актуальна, если вероятность срабатывания риска мала и последствия от его срабатывания незначительны. То есть не предпринимать никаких действий получается выгоднее и удобнее.

Для позитивных рисков стратегии работы в принципе аналогичны, но направлены на противоположные действия – повысить вероятность срабатывания рисков и усилить влияние последствий в результате срабатывания.

Стратегии управления для отрицательных рисков включают:

1 Принятие. Никаких усилий для борьбы с риском не предпринимается. Организация принимает возможность возникновения риска.

2 Передачу. Ответственность за устранение риска и его возможных последствий переходит к третьей стороне.

3 Избежание. Организация принимает меры для обеспечения того, чтобы риск не мог возникнуть.

4 Смягчение. Организация принимает меры по снижению вероятности возникновения риска или возможных негативных последствий его возникновения.

Для положительных рисков принятие также является жизнеспособной стратегией. Рассмотрим другие стратегии управления для положительных рисков:

1 Доля. Работа с третьей стороной будет происходить, чтобы увеличить, вероятно, положительный результат, и третья сторона согласиться разделить выгоды.

2 Повышение. Организация принимает меры для повышения вероятности возникновения риска и потенциальной выгоды в случае возникновения риска.

3 Использование. Организация работает, чтобы гарантировать, что событие действительно произойдет.

4 Предотвращение.

Вопросы данной стратегии управления:

а) можно ли сделать что-то заранее для уменьшения вероятности риска или его угрозы?

б) какой уровень вероятности или угрозы приемлем, чтобы считать работу по предотвращению риска достаточной?

Результаты вносятся в план:

а) перечень заранее проводимых мер для уменьшения вероятности и угрозы риска;

б) сроки каждого мероприятия, ответственный.

5 Избежание.

Вопросы данной стратегии управления:

а) можем ли мы избежать риска, изменив способ действия?

б) можем ли мы избежать риска, изменив что-то в требованиях к проекту (технологиях и т. д.)?

Результаты вносятся в план:

а) список изменений в проекте с обоснованием;

б) мероприятия для инициирования работ по списку со сроками исполнения.

6 Перенос риска.

Вопрос данной стратегии управления: можем ли мы перенести риск на другой проект, проектную группу, организацию или частных лиц?

Результаты вносятся в план:

а) рекомендуемые меры по переносу риска с обоснованием и указанием последствий (в том числе рискованных);

б) мероприятия для инициирования работ по рекомендуемым мерам со сроками;

в) действия при отказе от переноса риска.

7 Принятие.

Вопросы данной стратегии управления:

а) можем ли мы пережить последствия риска, если они все-таки наступят?

б) можем ли мы принять риск и не предпринимать по этому поводу никаких дальнейших действий?

Результат вносится в план: указание подхода с обоснованием выбора этого подхода.

8 Смягчение последствий.

Вопрос данной стратегии управления: может ли угроза риска быть уменьшена путем планирования некоторой реакции на него?

Результаты данной стратегии управления вносятся в план:

а) запасной план на случай возникновения проблемы (со сроками и ответственными);

б) триггеры (условия применения плана);

в) маркеры успеха предпринятых действий.

Качество – это целостная совокупность характеристик объекта, относящихся к его способности удовлетворять установленные или предполагаемые требования (например, требования технического задания).

Принципы качества всеобщего управления качеством (Total Quality Management, TQM):

1) обеспечение качества создаваемого продукта/результата проекта;

2) обеспечение качества планирования, разработки проектной документации;

3) качество выполнения работ по проекту в соответствии с плановой документацией;

4) качество ресурсного (и сырьевого) обеспечения проекта во время всего его жизненного цикла.

Процессы управления качеством проекта:

1 **Планирование качества** – процесс определения требований и/или стандартов качества для проекта и продукта, а также документирования того, каким образом проект будет демонстрировать соответствие установленным требованиям и стандартам.

Инструменты и методы:

1) сравнительный анализ затрат и выгод;

2) контрольные карты;

3) бенчмаркинг;

4) планирование экспериментов;

5) выборочные оценки;

6) разработка блок-схем.

Результаты планирования качества:

1) план управления качеством:

а) описывает, каким образом команда управления проектом будет претворять политику исполняющей организации в области качества;

б) является частью или вспомогательным планом в составе плана управления проектом;

2) метрики качества:

а) параметры проекта/продукта в конкретных терминах (фактические величины);

б) способы измерения этих параметров;

3) контрольные списки качества – это структурированный документ, обычно относящийся к конкретному элементу, который используется для подтверждения выполнения всех намеченных действий;

4) план совершенствования процессов;

5) обновления документов проекта.

2 **Обеспечение качества** – процесс проверки соблюдения требований к качеству и результатов измерений в процессе контроля качества для обеспечения применения соответствующих стандартов качества и оговоренных требований.

Инструменты и техники:

1) аудит качества:

а) проверка соответствия деятельности по реализации проекта принятым стандартам управления;

б) контроль исполнения корпоративных процедур управления и правильности оформления документов проекта;

в) подготовка рекомендаций по улучшению качества;

2) анализ процессов/экспертиза:

а) детальный анализ определенных областей деятельности в рамках проекта;

б) составление общей картины в целях повышения качества выполнения как данного проекта, так и проектов компании в целом.

Результаты обеспечения качества:

1) улучшение качества (совершение действий по увеличению эффективности и производительности процесса);

2) обновление плана управления проектом;

3) обновление документов проекта.

3 **Контроль качества** – процесс контроля и записи результатов выполнения действий по обеспечению качества для оценки исполнения и разработки рекомендаций относительно необходимых изменений.

Инструменты и методы:

1) инспекция (измерения, испытания, тестирования, чтобы удостовериться, что результат удовлетворяет требованиям);

2) диаграммы Ишикавы, Парето;

3) метод «Пять почему»;

4) анализ трендов.

Результаты контроля качества:

1) заполненные проверочные списки;

2) корректирующие действия;

3) настройка процесса;

4) улучшение качества.

Практическое задание:

1 Составить структуру разбиения работ вашего проекта.

2 Составить диаграмму Ганта вашего проекта.

3 Определить риски, связанные с требованиями. Результаты внести в таблицу, образец которой представлен таблицей 5.2.

4 Оформить отчет и защитить лабораторную работу.

Таблица 5.2 – Возможные риски

Первопричина	Условия	Последствия	Приносимый ущерб

Содержание отчета:

1 Цель работы.

2 Структура разбиения работ.

3 Диаграмма Ганта.

4 Таблица возможных рисков сбора и анализа требований.

5 Выводы по работе.

Контрольные вопросы:

1 Что такое план?

2 Что такое метод критического пути?

3 Что такое риск?

4 Что такое критический путь?

5 Как производится оценка риска?

6 Опишите стратегию управления рисками.

Лабораторная работа №6
Коммуникация IT-проекта

Цель: изучить подходы к коммуникации с заинтересованными сторонами в зависимости от специфики проекта и иных факторов, научиться составлять план коммуникаций по проекту.

План занятия:

1 Изучить теоретические сведения.

2 Выполнить практическое задание по лабораторной работе.

3 Оформить отчет и ответить на контрольные вопросы.

Теоретические сведения

Заинтересованная сторона (ЗС) – любая группа (или индивид), способная влиять или влияющая на достижение цели организации, на способность предприятия приносить доход, а также все бенефициары и источники риска.

ЗС могут выполнять операции в разное время жизненного цикла проекта и с разным уровнем глубины и детализации. Существует большое количество подходов и классификаций заинтересованных сторон.

Заинтересованные лица – это те, кто:

- выдвигает требования;
- вовлечен в работу над проектом или осуществляет какие-либо действия в ходе его реализации;
- несет ответственность перед организацией за успешность проекта;
- налагает ограничения на осуществление проекта или инициативы;
- чувствует на себе влияние результатов проекта.

Участники проекта – физические лица и организации, которые непосредственно вовлечены в проект и чьи интересы могут быть затронуты при осуществлении проекта.

Клиент – это человек (или организация), получающий от продукта прямую или косвенную выгоду.

Процесс анализа заинтересованных сторон состоит из следующих этапов:

- определить все потенциальные заинтересованные стороны проекта и существующую информацию о них, такую как роли, отделы, интересы, уровни знаний, ожидания и уровни влияния;
- определить степень потенциального влияния и поддержки, которые могут оказать каждая из заинтересованных сторон проекта, и классифицировать их таким образом, чтобы можно было найти подход к ним;
- оценить, каким образом ключевые заинтересованные стороны проекта скорее всего будут реагировать или действовать в разнообразных ситуациях для того, чтобы спланировать, как повлиять на них с целью усиления их поддержки и сокращения потенциальных отрицательных влияний.

Анализ ЗС проекта позволяет понять, у каких ЗС стоит собирать требования к результатам проекта; какие ЗС стоит рассматривать как источники рисков для проекта; какие ЗС учесть в плане коммуникаций проекта (в плане коммуникаций нужно описать, какую информацию, в каком виде и как часто нужно высылать каждой из ЗС); какие дополнительные работы следует включить в список работ по проекту для управления заинтересованными сторонами проекта.

Рассмотрим матрицу власти/интересов как способ анализа заинтересованных сторон (рисунок 6.1).

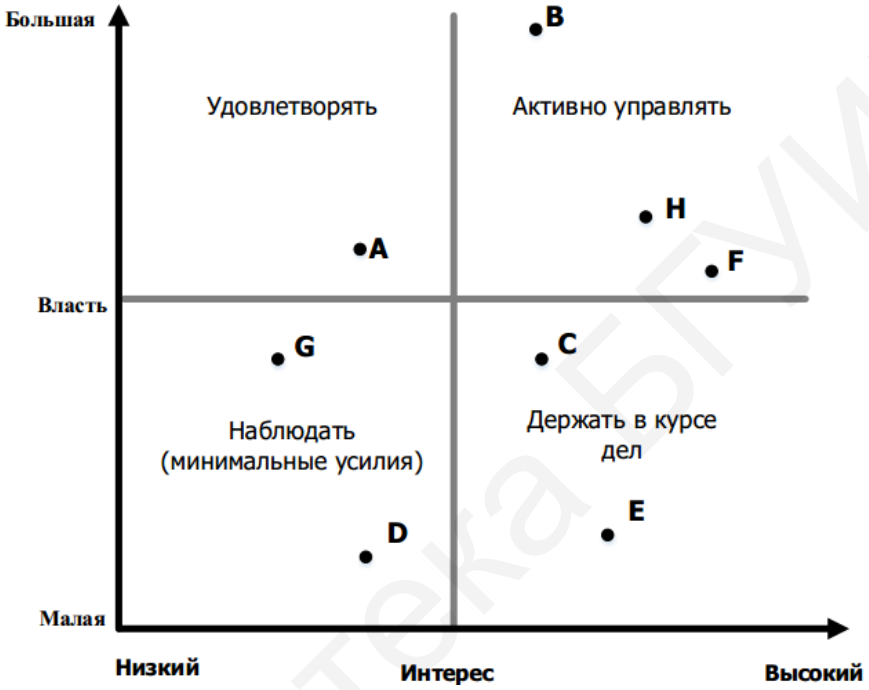


Рисунок 6.1 – Матрица власти/влияния

Расположение ЗС на матрице:

1 Наблюдать. Периодически уточнять, не усилился ли у ЗС интерес к проекту или не повысился ли уровень власти, т. к. изменение по одной из этих шкал приведет к перемещению в другой квадрант матрицы и к необходимости изменить стратегию работ.

2 Держать в курсе дел. Всячески информировать о будущих результатах проекта, о ходе работ над проектом, об изменениях в проекте для формирования позитивного отношения к проекту.

3 Удовлетворять. Так как ожидания от проекта у этой группы невелики, то, как правило, их достаточно легко реализовать в проекте. Потому нужно выяснить ожидания от проекта, перевести их в требования к результатам проекта и реализовать эти требования. Если вдруг ожидания не совпадают с целями проекта, следует объяснить ЗС, что в данном проекте эти ожидания не предполагается реализовывать.

4 Активно управлять. Вовлекать в принятие решений по проекту, часто информировать о выполнении принятых решений, демонстрировать промежуточные результаты работ по проекту.

Очень важно периодически повторять анализ заинтересованных сторон в ходе реализации проекта (интересы и власть у ЗС могут меняться по ходу проекта, могут добавляться новые заинтересованные стороны, которых на старте проекта не удалось идентифицировать).

Важным фактором достижения успеха проекта является выявление информационных потребностей участников проекта и определение подходящих средств удовлетворения этих потребностей.

Планирование коммуникаций – определение потребностей участников проекта в коммуникации и информации.

В плане должны быть представлены:

- способ предоставления информации;
- целевая аудитория;
- время и частота коммуникаций.

Пример плана коммуникации представлен в таблице 6.1.

Таблица 6.1 – Примерный план коммуникации

Заинтересованное лицо (должность, ФИО, краткая информация)	Область ответственности/ влияния (предмет коммуникации)	Какой документ создает (цель)	Средство и частота коммуникации (как, когда)

Рассчитать количество информационных каналов можно по формуле

$$N = n \cdot \frac{(n-1)}{2}, \quad (6.1)$$

где n – число заинтересованных лиц.

Распространение информации – своевременное предоставление необходимой информации участникам проекта.

Выделяют следующие каналы коммуникации:

1) устный/письменный;
2) формальный/неформальный;
3) внешний/внутренний.

Барьеры коммуникации:

а) расстояние;
б) шумы;
в) язык;
г) время;
д) амбиции;
е) сокрытие информации;

ж) культурные различия;

з) использование неверного канала получения и отправки сообщений.

Критерии выбора способа коммуникации:

1) география;

2) культура;

3) отношение:

а) ко времени;

б) к завершению задач;

в) к контрактам;

г) к формальной и фактической власти;

4) тип проекта:

а) новый проект;

б) кастомизированный внутренний проект;

5) частота взаимодействия;

6) степень формальности.

При выборе способа коммуникации следует учитывать:

- где находится собеседник или аудитория;

- подход;

- типы коммуникации (status, anomalies, issues/resolution, risks, meeting results, action items);

- результат (какие требования и каков способ их выполнения);

- как быстрее донести информацию до определенной аудитории (signoff authority, veto authority, or review only);

- ресурсные и временные ограничения.

Отчетность по исполнении – сбор и распространение информации о выполнении работ. Этот процесс включает в себя:

а) сбор всех данных базового плана;

б) информацию о выполнении работ (о содержании, сроках, стоимости, качестве, рисках и поставках);

в) предоставление этой информации участникам проекта (совещание по оценке текущего состояния).

Управление участниками проекта – управление коммуникациями в целях удовлетворения требований участников и решения возникающих проблем.

Практическое задание:

1 Составить план коммуникаций по вашему проекту. Результаты внести в таблицу, образец которой представлен таблицей 6.2.

Таблица 6.2 – План коммуникаций

Заинтересованная сторона	Цель коммуникации	Формат и способ коммуникации	Результат

2 Составить матрицу RACI для любых четырех заинтересованных лиц проекта по любым пяти функциям (R (Responsible) – выполняет работу, A (Accountable) – несет ответственность за выполнение работы, C (Consult) – консультирует по выполнению работы, I (Informed) – информируется о ходе работ). Результаты внести в таблицу, образец которой представлен таблицей 6.3.

3 Оформить отчет и защитить лабораторную работу.

Таблица 6.3 – Матрица RACI

Функция	Заинтересованная сторона			
	1	2	3	4

Содержание отчета:

1 Цель работы.
2 План коммуникаций.
3 Матрица RACI.
4 Выводы по работе.

Контрольные вопросы:

1 Кто такой клиент?
2 Кто такой участник проекта?
3 Кто является заинтересованной стороной?
4 Является ли клиент заинтересованной стороной?
5 Как матрица влияния помогает анализировать ЗС?
6 Что такое планирование коммуникаций?
7 Что такое распространение информации?
8 Что такое отчетность по коммуникации?

Список использованных источников

1 Информационный ресурс СООО «Образовательный центр Парка высоких технологий», 2016–2017 гг.

2 Вигерс, К. Разработка требований к программному обеспечению / К. Вигерс ; пер. с англ. – 3-е изд., доп. – М. : Русская редакция ; СПб. : БХВ-Петербург, 2014. – 736 с.

3 Управление проектами – понятия и процессы [Электронный ресурс]. – 2018. – Режим доступа : http://www.myshared.ru/slide/105617.

4 Дитхелм, Г. Управление проектами. В 2 т. Т. 1 / Г. Дитхелм ; пер. с нем. – СПб. : Изд. дом «Бизнес-пресса», 2004. – 400 с.

5 Беркун, С. Искусство управления IT-проектами / С. Беркун. – СПб. : Питер, 2010. – 432 с.

6 Лекции по управлению программными проектами [Электронный ресурс]. – 2018. – Режим доступа : https://ita.sibsutis.ru/sites/csc.sibsutis.ru/files/courses/trpo/sw_project_management.pdf.

7 Полковников, А. В. Управление проектами. Полный курс MBA / А. В. Полковников, М. Ф. Дубовик. – М. : ЗАО «Олимп-Бизнес», 2015. – 552 с.

8 Курбацкий, В. Н. Разработка и управление проектами средствами Microsoft Project 2010 / В. Н. Курбацкий, С. И. Максимов. – Минск : РИВШ, 2012. – 90 с.

9 Kerzner, H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling / H. Kerzner. – John Wiley & Sons, Inc., 2003. – 914 p.

10 Руководство к своду знаний по управлению проектами [Электронный ресурс]. – 2018. – Режим доступа : http://sfpk.at.ua/biblioteka/PMI/pmbok2004_rus.pdf.

11 Требования к содержанию документов [Электронный ресурс]. – 2018. – Режим доступа : http://quantech.ru/upload/iblock/42c/rd-50_34.698.90.pdf.

12 Виды, комплектность и обозначение документов при создании автоматизированных систем [Электронный ресурс]. – 2018. – Режим доступа : http://clc.to/R-DrRg.