

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное
автономное образовательное
учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ РАБОТАМ

«Проектная деятельность»

Направление подготовки 09.03.02 «Информационные системы и
технологии»

Направленность (профиль) «Прикладное программирование и интернет-
технологии»

Квалификация выпускника: бакалавр

Ставрополь

2026

Введение

Цель изучения дисциплины «Проектная деятельность» формирование системного методического подхода к проектной деятельности и приобретение практических навыков проектной работы IT-области, формирование высокой проектной культуры, а также компетенций будущего бакалавра по направлению подготовки 09.03.02 Информационные системы и технологии.

Задачами обучения являются:

- усвоение роли грамотной организации проектной деятельности для эффективного решения проектных задач различной сложности;
- изучение основ и методов планирования этапов будущего проекта;
- изучение основ управления в проектной деятельности;
- обретение навыков формирования и формулирования задач для индивидуальной и совместной (коллективной) проектной деятельности;
- применение инновационных креативных технологий и методик для создания и совершенствования творческих идей;
- обретение навыков правильного оформления готового проекта для презентации (в том числе, заказчику), для выставки, просмотра, печати, архива.

1. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
УК-2 – способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2 _{ид-1.1} - формулирует цель проекта, определяет совокупность взаимосвязанных задач, обеспечивающих ее достижение и определяет ожидаемые результаты решения задач. УК-2 _{ид-1.2} - разрабатывает план действий для решения задач проекта, выбирая оптимальный способ их решения, исходя из действующих	Результаты демонстрации знаний о грамотно сформулированной цели проекта. Верно определена совокупность взаимосвязанных задач, обеспечивающих ее достижение и определяет ожидаемые результаты

	правовых норм и имеющихся ресурсов и ограничений УК-2_{ид-1.3} - обеспечивает выполнение проекта в соответствии с установленными целями, сроками и затратами, исходя из действующих правовых норм, имеющихся ресурсов и ограничений, в том числе с использованием цифровых инструментов	решения задач. Результаты применения современных подходов для разработки плана действий для решения задач проекта с помощью оптимальных способов решения, исходя из действующих правовых норм и имеющихся ресурсов и ограничений являются верными. Результаты выполнения проекта в соответствии с установленными целями, сроками и затратами, исходя из действующих правовых норм, имеющихся ресурсов и ограничений, в том числе с использованием цифровых инструментов.
УК-3 – способен осуществлять социальное взаимодействие и реализовывать свою роль в команде	УК-3_{ид-3.1} участвует в межличностном и групповом взаимодействии, используя инклюзивный подход, эффективную коммуникацию, методы командообразования и командного взаимодействия при совместной работе в рамках поставленной задачи. УК-3_{ид-3.2} – обеспечивает работу команды для получения оптимальных результатов совместной работы, с учетом индивидуальных возможностей её членов, использования методологии достижения успеха, методов, информационных технологий и технологий форсайта. УК-3_{ид-3.3} – обеспечивает выполнение поставленных задач на основе мониторинга командной работы и своевременного реагирования на существенные отклонения	Проводит предпроектное обследование предметной области, выявляет бизнес-проблему для разработки проекта. На основе предпроектного обследования формирует цели и задачи проекта, формирует концепцию проекта Разрабатывает техническое задание на основе ГОСТ разрабатывает план реализации проекта и его этапов, формирует требования к разработке, устанавливает виды работ и их сроки. Использует CASE-системы для проектирования, управления проектом и его ресурсами, составляет график работ, использует методы предпроектного обследования и проектирования.
УК-6 – способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни	УК-6_{ид-6.1} – устанавливает личные и профессиональные цели в соответствии с уровнем своих ресурсов и приоритетов действий, для успешного развития в избранной сфере профессиональной деятельности УК-6_{ид-6.2} – реализует и корректирует стратегию личностного и профессионального развития, с учетом условий, средств, личностных	Анализирует и оценивает риски при проведении проектов. Определяет цели и задачи проекта, формирует его концепцию и план. Планирует и контролирует выполнение работ по проекту.

	возможностей, этапов карьерного роста, временной перспективы развития деятельности и требований рынка труда УК-6_{ид-6.3} – критически оценивает эффективность использования времени и других ресурсов при решении поставленных задач в избранной сфере профессиональной деятельности	
УК-9 – Способен принимать обоснованные экономические решения в различных областях жизнедеятельности	УК-9_{ид-9.1} - понимает базовые принципы функционирования экономики и экономического развития, цели и формы участия государства в экономике УК-9_{ид-9.2} – применяет методы личного экономического и финансового планирования для достижения текущих и долгосрочных финансовых целей УК-9_{ид-9.3} – использует финансовые инструменты для управления личными финансами, контролирует собственные экономические и финансовые риски	Управляет бюджетом и ресурсами проекта. Разрабатывает стратегии управления изменениями в рамках проектов. Анализирует и оценивает влияние социальных, экономических и экологических факторов на проектную деятельность

Практическая работа № 1.

Предпроектное обследование

Цель: ознакомиться с технологиями предпроектного обследования.

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

Целью предпроектного обследования является преобразование общих нечетких знаний о предназначении будущей информационной системы в сравнительно точные требования к нему. Существуют два варианта неопределенности:

- неизвестны методы решения формулируемой задачи - такого типа не определенности обычно возникают при решении научно-технических задач;
- неизвестна структура автоматизируемых информационных процессов - обычно встречается при построении автоматизированных систем управления предприятиями.

В первом случае во время предпроектных обследований определяют возможность решения поставленной задачи и методы, позволяющие получить требуемый результат, что может потребовать соответствующих научных исследований как фундаментального, так и прикладного характера, разработки и исследования новых моделей объектов реального мира.

Во втором случае определяют:

- структуру и взаимосвязи автоматизируемых информационных процессов;
 - распределение функций между человеком и системой, а также между аппаратурой и программным обеспечением;
 - функции программного обеспечения; внешние условия его функционирования и особенности его интерфейсов, как с пользователями, так и при необходимости - с аппаратной частью;
 - требования к программным и информационным компонентам, необходимые аппаратные ресурсы, требования к базам данных и физические

характеристики программных компонент.

Результаты предпроектных обследований предметной области используют в процессе разработки технического задания.

Предпроектное обследование обычно состоит из трех этапов:

- 1) предварительное обследование (сбор сведений об объекте);
- 2) анализ сведений (описание и моделирование предметной области);
- 3) оценка эффективности и целесообразности ИТ-проекта.

На каждом этапе применяются различные методы исследования систем управления, контроля и/или мониторинга. Цели ИС: улучшение, изменение (реинжиниринг) или автоматизация бизнес-процессов.

Предмет исследования, то есть то, что изучает исследователь с целью решения проблем на объекте, это: система знаний, умений и навыков; методы и способы; факторы внешней и внутренней среды; процессы, происходящие в организации.

Объектом исследования является организация и ее подсистемы – реальные физические объекты, измеряемые качественными и количественными показателями.

Научное исследование, как правило, проводится в предметных рамках определенного научного подхода с использованием группы научных методов.

Цели исследования системы управления: улучшение, изменение (реинжиниринг) или автоматизация систем управления.

Метод исследования – это способ получения нового знания, а также инструментарий, с помощью которого проводится исследование; совокупность приемов обработки информации, позволяющих достичь целей исследования.

Выбор методов исследования, интеграция различных методов при проведении предпроектного обследования определяется знанием, опытом и интуицией специалистов, проводящих исследование.

Методы исследования предметной области проектирования ИС можно разбить на следующие группы:

основанные на использовании знаний и интуиции специалистов (экспертные методы);

формализованного представления систем, основанные на использовании математических, экономико-математических методов и методов моделирования, среди которых при проектировании информационных систем особое место занимают графические, включающие различные методы графического представления информации.

комплексированные – сформировались путем интеграции экспертных и формализованных методов (комбинаторика, ситуационное моделирование, топология, графосемиотика и др).

На предварительном этапе обследования решаются следующие задачи: предварительное выявление требований к будущей системе;

- определение структуры организации;
- определение перечня целевых функций организации;
- анализ распределения функций по подразделениям и сотрудникам;
- выявление функциональных взаимодействий между подразделениями, информационных потоков внутри подразделений и между ними, внешних информационных воздействий;
- анализ существующих средств автоматизации организации и др.

Длительность предварительного обследования обычно составляет 1-2 недели. В течение этого времени системный аналитик должен обследовать не более 2-3 видов деятельности (учет кадров, бухгалтерия, перевозки, маркетинг и др.). Сбор информации для построения полной бизнес-модели организации часто сводится к изучению документированных информационных потоков и функций подразделений, а также производится путем интервьюирования и анкетирования.

К началу работ по обследованию организация обычно предоставляет комплект документов, в состав которого обычно входят:

- 1) сводная информация о деятельности предприятия (оргструктура, информация об управленческой, финансово-экономической, производственной

деятельности предприятия, сведения об учетной политике и отчетности);

2) регулярный документооборот предприятия (реестры входящей, исходящей и внутренней информации (таблица 1.);

3) сведения об инфраструктуре предприятия;

4) сведения об ответственных лицах и исполнителях.

Таблица 1 – Реестры информации

Реестр входящей информации						
(Наименование предприятия)		(Наименование подразделения)		Характеристики обработки документов		
№	Наименование и назначение документа	Кто обрабатывает	Откуда поступает	Трудоемкость	Периодичность, регламент	Способ получения
Реестр внутренней информации						
(Наименование предприятия)		(Наименование подразделения)		Характеристики обработки документов		
№	Наименование и назначение документа	Кто обрабатывает	Кому передает	Трудоемкость	Периодичность, регламент	Способ получения
Реестр исходящей информации						
(Наименование предприятия)		(Наименование подразделения)		Характеристики обработки документов		
№	Наименование и назначение документа	Кто обрабатывает	Куда поступает	Трудоемкость	Периодичность, регламент	Способ получения

Один из методов исследования, который может быть применен на этапе сбора сведений – интервьюирование. Списки вопросов для интервьюирования и анкетирования составляются по каждому обследуемому подразделению и утверждаются руководителем компании.

Это делается с целью:

- предотвращения доступа к конфиденциальной информации;
- усиления целевой направленности обследования;
- минимизации отвлечения сотрудников предприятий от выполнения должностных обязанностей.

Общий перечень вопросов (с их последующей детализацией) включает следующие пункты:

- основные задачи подразделений;

- собираемая и регистрируемая информация;
- отчетность;
- взаимодействие с другими подразделениями.

Анкететы для руководителей и специалистов могут содержать следующие вопросы:

1. Каковы (с позиций вашего подразделения) должны быть цели создания информационной системы;
2. Организационная структура подразделения;
3. Задачи подразделения;
4. Последовательность действий при выполнении задач;
5. С какими типами внешних организаций (банк, заказчик, поставщик и т.п.) взаимодействует подразделение и какой информацией обменивается;
6. Каким справочным материалом вы пользуетесь;
7. Сколько времени (в минутах) вы тратите на исполнение основных операций; на какие даты приходятся «пиковые нагрузки» (периодичность в месяц, квартал, год и т.д.); техническое оснащение подразделения (компьютеры, сеть, модем и т.п.); используемые программные продукты для автоматизации бизнес-процессов;
8. Какие отчеты и как часто вы готовите для руководства; ключевые специалисты подразделения, способные ответить на любые вопросы по бизнес-процессам, применяемым в подразделении;
9. Характеристики удаленных объектов управления;
10. Документооборот на рабочем месте.

Собранные таким образом данные, как правило, не охватывают всех существенных сторон организационной деятельности и обладают высокой степенью субъективности. И самое главное, что такого рода обследования не выявляют устойчивых факторов, связанных со специфическими особенностями

организации, воздействовать на которые можно исключительно методами функциональной настройки организационной системы. Анализ опросов руководителей обследуемых организаций и предприятий показывает, что их представления о структуре организации, общих и локальных целях функционирования, задачах и функциях подразделений, а также подчиненности работников иногда имеют противоречивый характер. Кроме того, эти представления подчас расходятся с официально декларируемыми целями и правилами или противоречат фактической деятельности. Если структуру информационных потоков можно выявить по образцам документов и конфигурациям компьютерных сетей и баз данных, то структура реальных микропроцессов, осуществляемых персоналом в информационных контактах (в значительной мере недокументированных) остается неизвестной. Ответы на эти вопросы может дать структурно-функциональная диагностика, основанная на методах сплошной (или выборочной) фотографии рабочего времени персонала.

Цель диагностики – получение достоверного знания об организации и организационных отношениях ее функциональных элементов. В связи с этим к важнейшим задачам функциональной диагностики организационных структур относятся:

- классификация субъектов функционирования (категорий и групп работников);
- классификация элементов процесса функционирования (действий, процедур);
- классификация направлений (решаемых проблем), целей функционирования;
- классификация элементов информационных потоков;
- проведение обследования деятельности персонала организации;
- - исследование распределения (по времени и частоте) организационных характеристик: процедур, контактов персонала, направлений деятельности, элементов информационных потоков – по отдельности и в комбинациях друг с другом по категориям работников, видам процедур и их направлениям (согласно результатам и логике исследований);

- выявление реальной структуры функциональных, информационных, иерархических, временных, проблемных отношений между руководителями, сотрудниками и подразделениями;
- установление структуры распределения рабочего времени руководителей и персонала относительно функций, проблем и целей организации;
- выявление основных технологий функционирования организации (информационных процессов, включая и недокументированные), их целеполагания в сравнении с декларируемыми целями организации;
- выявление однородных по специфике деятельности, целевой ориентации и реальной подчиненности групп работников, формирование реальной модели организационной структуры и сравнение ее с декларируемой;
- определение причин рассогласования декларируемой и реальной структуры организационных отношений.

Сплошной «фотографией» рабочего времени называется непрерывное наблюдение и регистрация характеристик работников в процессе функционирования в течение всего рабочего дня. Информация последовательно вносятся в заранее заготовленную рабочую таблицу, которая по окончании процедуры обследования пополняется дополнительными характеристиками: технологическая ветвь, системная функция, предмет, аспект, эмоциональный фон и др. Часть показателей заполняются в процессе обследования, остальные – после. Содержание рабочей таблицы может быть следующим:

1. Номер (по порядку);
2. Агент (должность обследуемого работника);
3. Время, в течение которого выполнялась процедура;
4. Процедура (наименование содержания совокупности элементарных действий, объединенных общностью решаемой частной задачи);
5. Содержание (суть процедуры, которая должна быть классифицирована);
6. Информация (направление движения информации между агентом и

контрагентом);

7. Инициатива (инициатор начала выполнения данной процедуры);
8. Контрагент (должность работника, который находится с обследуемым в контакте);
9. Отношение (отражающая субординацию агента и контрагента форм взаимодействия в данной процедуре);
10. Проблема (словесная характеристика решаемой проблемы).

Результатом предпроектного обследования должен явиться «Отчет об экспресс- обследовании предприятия», который включает следующее:

1. Краткое схематичное описание бизнес-процессов (например: управление закупками и запасами, управление производством, управление продажами, управление финансовыми ресурсами);
2. Основные требования и приоритеты автоматизации;
3. Оценка необходимых для обеспечения проекта ресурсов заказчика;
4. Оценка возможности автоматизации, предложения по созданию автоматизированной системы с оценкой примерных сроков и стоимости.

Документы, входящие в отчет об обследовании, могут быть представлены в виде текстового описания или таблиц с описанием операций, исполнителей и документов бизнес-процессов. Информация, полученная в результате предпроектного обследования, анализируется с помощью методов структурного и/или объектного анализа, и используется для построения моделей деятельности организации.

Модель организации предполагает построение двух видов моделей:

- модели «как есть», отражающей существующее на момент обследования положение дел в организации и позволяющей понять, каким образом функционирует данная организация, а также выявить узкие места и сформулировать предложения по улучшению;
- модели «как должно быть», отражающей представление о новых

технологиях работы организации.

Каждая из моделей включает в себя полную функциональную и информационную модель деятельности организации, а также модель, описывающую динамику поведения организации (в случае необходимости). В качестве основного каркаса, объединяющего и систематизирующего все знания по бизнес-модели, можно использовать различные эталонные (референтные) модели.

Методика и порядок выполнения работы

Задание 1. Изучить стандарты ГОСТ 19.104-78 ЕСПД, ГОСТ 24.301-80

Задание 2. Изучить методы предпроектного исследования.

Задание 3. Составить анкету опроса.

Задание 4. Составить реестр входных и выходных данных

Задание 5. Разработать отчёт о предпроектном обследовании предметной области (по индивидуальному варианту) для внедрения в фирме/организации Информационной системы. Оформление отчета должно соответствовать требованиям стандартов.

ГОСТ 19.104-78 ЕСПД. Основные надписи» по оформлению листа утверждения и титульного листа, ГОСТ 24.301-80 Система технической документации на АСУ. Общие требования к выполнению текстовых документов» по оформлению остальной части документа.

Отчет должен содержать лист утверждения, титульный лист, пояснительную записку (2-3 л) предпроектного обследования (схематическое описание бизнес- процессов, требования к системе).

Индивидуальные задания

№ варианта	Задание
1	Разработка подсистемы учета товаров интернет-магазина
2	Разработка подсистемы учета и обработки заявок ремонтных работ ЖКХ
3	Разработка подсистемы отдела продаж магазина
4	Разработка подсистемы учета канцтоваров отдела
5	Разработка подсистемы телефонного справочника

6	Разработка подсистемы учета транспортных средств предприятия
7	Разработка подсистемы поставки товара в магазин
8	Разработка подсистемы учета доставки товара
9	Разработка подсистемы оформления заказа интернет-магазина
10	Разработка подсистемы учета книг библиотеки

Содержание отчета и его форма

Представить отчёт в формате .doc о предпроектном обследовании фирмы/организации (по индивидуальному варианту) для разработки информационной системы (подсистемы).

Вопросы для защиты работы

1. Перечислите основные этапы предпроектного обследования
2. Дайте определение понятию объекта исследования
3. Дайте определение понятию предмета исследования
4. Дайте определение понятию метода исследования
5. Какие группы методов исследования предметной области проектирования ИС вам известны? Дайте им краткую характеристику
6. Какие решаются задачи на предварительном этапе обследования?
7. Как производится сбор информации на предпроектном обследовании?
8. Какова роль анкет руководителей и специалистов в предпроектном обследовании?
9. В чем заключается цель диагностики?
10. 10.Перечислите задачи функциональной диагностики
- 11.Что называется сплошной «фотографией» рабочего времени?
- 12.Что является результатом предпроектного обследования?
- 13.Каково содержание отчета об экспресс-обследовании?
- 14.Какие модели выстраиваются в предпроектном обследовании?
15. Какие структурные данные входят в листы утверждения и титульного листа в программных документах?
16. Перечислите основные надписи листа утверждения.

17. Входит ли лист утверждения в общее количество листов документа?
18. На что распространяется стандарт 24.301-80?
19. Перечислите основной состав документов в соответствии с ГОСТ 24.301-80.

Практическая работа № 2.

Составление технического задания на разработку ИС

Цель: Ознакомиться с процедурой разработки технического задания на создание программного продукта (ПП) с применением ГОСТ 19.102-77 «Стадии разработки программ и программной документации» и ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Основные теоретические сведения

Техническое задание - это документ, определяющий цели, требования и основные исходные данные, необходимые для разработки автоматизированной системы управления.

Техническое задание представляет собой документ, в котором сформулированы основные цели разработки, требования к программному продукту, определены сроки и этапы разработки и регламентирован процесс приемо-сдаточных испытаний.

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

В основе этого документа лежат исходные требования заказчика, анализ передовых достижений техники, результаты выполнения научно-исследовательских работ, предпроектных исследований, научного прогнозирования и т.п.

При разработке технического задания (ТЗ) необходимо решить следующие задачи:

- установить общую цель создания АИС;
- установить общие требования к проектируемой системе;
- разработать и обосновать требования, предъявляемые к информационному, математическому, программному, техническому и технологическому обеспечению;
- определить состав подсистем и функциональных задач;

- разработать и обосновать требования, предъявляемые к подсистемам;
- определить этапы создания системы и сроки их выполнения;
- провести предварительный расчет затрат на создание системы и определить уровень экономической эффективности ее внедрения;
- определить состав исполнителей.

В Российской Федерации действует ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы», также на техническое задание существует стандарт ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению».

ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам устанавливает общие требования к оформлению программных документов. Программный документ должен состоять из следующих частей:

- Титульной;
- Информационной;
- Основной.

Титульная часть оформляется согласно ГОСТ 19.104-78 ЕСПД. Основные надписи.

Информационная часть должна состоять из аннотации и содержания. В аннотации приводят сведения о назначении документа и краткое изложение основной части.

Содержание включает перечень записей о структурных элементах основной части документа.

Состав и структура основной части программного документа устанавливается стандартами ЕСПД на соответствующие документы.

Основная часть технического задания должна содержать следующие разделы:

ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению:

1. введение;

2. основания для разработки;
3. назначение разработки;
4. требования к программному продукту;
5. требования к программной документации;
6. технико-экономические показатели;
7. стадии и этапы разработки;
8. порядок контроля и приёмки;

В зависимости от программного продукта допускается уточнять содержание разделов, объединять отдельные из них, вводить новые разделы. В техническое задание допускается включать приложения.

Задания для выполнения

1. Разработать техническое задание на программный в соответствии с ГОСТ19.201-78 и ГОСТ 34.602-89
2. Оформить работу в соответствии с ГОСТ 19.106-78. При оформлении использовать MS Office.
3. Сдать и защитить работу.

Содержание отчета и его форма

Представить отчёт в формате .doc о предпроектном обследовании фирмы/организации (по индивидуальному варианту) для разработки информационной системы (подсистемы).

Вопросы для защиты работы

1. Что такое техническое задание?
2. Каким гостом определяется ТЗ для разработки программного обеспечения?
3. Каким гостом определяется ТЗ для разработки автоматизированных информационных систем?
4. Какие разделы входят в содержание ТЗ.

5. Опишите содержание ТЗ

Практическая работа № 3.

Построение модели бизнес-процессов информационной системы

Цель и содержание работы: сформировать умение находить и описывать процессы в заданной предметной области. найти не менее 5 процессов, указать их основные составляющие: входы, выходы, поставщиков, клиентов, классифицировать их на первичные/вторичные, внутренние/внешние. Найти взаимосвязь процессов между собой, состыковать их по входам-выходам. Научиться строить модель бизнес-процессов информационной системы,

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

Бизнес-процесс (БП) определяется как логически заверченный набор взаимосвязанных и взаимодействующих видов деятельности, поддерживающий деятельность организации и реализующий ее политику, направленную на достижение поставленных целей.

Бизнес-процесс использует определенные ресурсы (финансовые, материальные, человеческие, информационные) для преобразования входных элементов в выходные. Важным шагом структуризации деятельности являются выделение и классификация бизнес-процессов.

Можно выделить следующие классы процессов:

- основные процессы;
- обеспечивающие процессы и процессы управления.

Основными бизнес-процессами являются процессы, непосредственно связанные с созданием стоимости, ориентированные на производство товаров или оказание услуг, составляющих основную деятельность и обеспечивающих получение дохода.

Обеспечивающие бизнес-процессы не увеличивают ценность продукта

или услуги для потребителя. Они предназначены для поддержки выполнения основных бизнес-процессов.

Таковыми процессами являются финансовое обеспечение деятельности, обеспечение кадрами, юридическое обеспечение, администрирование, обеспечение безопасности, поставка комплектующих материалов, ремонт и техническое обслуживание и т.д.

Бизнес-процессы управления - это процессы, охватывающие весь комплекс функций управления на уровне каждого бизнес-процесса и системы в целом. Примерами таких процессов могут быть процессы стратегического, оперативного и текущего планирования, процессы формирования и выполнения управляющих воздействий. Процессы управления оказывают воздействие на остальные процессы организации.

Бизнес-модель - это формализованное (в данном случае — графическое) описание процессов, связанных с ресурсами, и отражающих существующую или предполагаемую деятельность предприятия.

На рисунке приведен пример бизнес-процесса в виде блок-схемы.

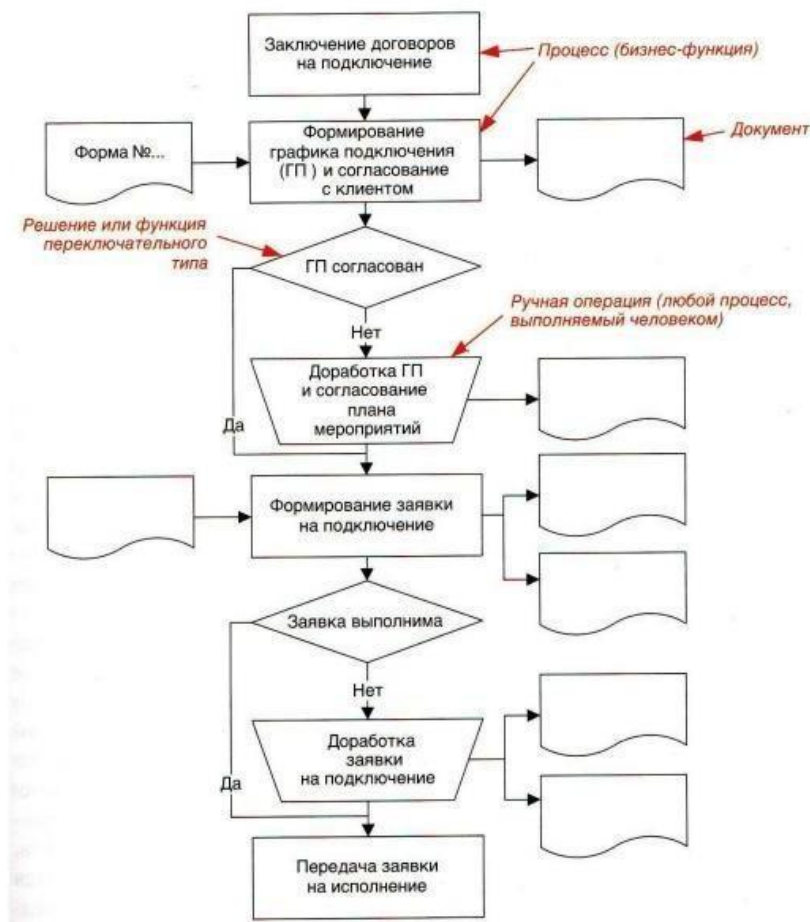


Рисунок 1 – Пример модели бизнес-процесса

Построение бизнес-моделей заключается в применении различных методов и средств для визуального моделирования бизнес-процессов.

Цели моделирования:

- обеспечить понимание структуры информационной системы (ИС) или организации и динамики происходящих в ней процессов;
- обеспечить понимание текущих проблем и возможностей их решения;
- убедиться, что заказчики, пользователи и разработчики одинаково понимают цели и задачи организации;
- создать базу для формирования требований к будущей ИС организации. БП можно рассматривать с разных точек зрения.

БП как деятельность:

- работа «от начала до конца»;
- поток работы, проходящий от одного специалиста к другому или от одного отдела к другому;
- взаимонезависимый компонент производственной системы, преобразующие вход в один или несколько выходов в соответствии с предварительно установленными правилами;
- одна или более связанных между собой процедур или операций (функций), которые совместно реализуют некую бизнес-задачу или политическую цель предприятия.

БП как создание продукта/услуги:

- множество внутренних шагов деятельности, начинающихся с одного и более входов и заканчивающихся созданием продукции, необходимой клиенту;
- связанный набор повторяемых действий (функций), которые преобразуют исходный материал и/или информацию в конечный продукт (услугу) в соответствии с определенными критериями.

БП как формирование прибавочной и/или потребительной стоимости: совокупность различных видов деятельности, в рамках которой “на входе” используется один или более видов ресурсов, и в результате этой деятельности “на выходе” реализуется товар, представляющий ценность для потребителя.

Определение бизнес-процесса:

- Название (определение) процесса;
- Реализуемая функция или их последовательность;
- Участники процесса;
- Владелец процесса;
- Входные и выходные потоки, а так же их поставщики (или потребители);
- Требуемые ресурсы (производственные, технические, материальные, информационные);
- Границы процесса



- Определяющая цель (цели) процесса;
- Метрики процесса, точки и процедуры мониторинга процесса;
- Возможные риски и влияния процесса на субъектов процесса; Документ - описание процесса.

Семь свойств процесса:

1. Результативность – характеризует соответствие результатов процесса нуждам и ожиданиям потребителей;
2. Определенность – отражает степень, с которой реальный процесс соответствует описанию;
3. Управляемость – характеризует степень, в которой производится управление выполнением процесса производства требуемых продуктов/услуг, отвечающих определенным целевым показателям;
4. Эффективность – отражает, насколько оптимально используются ресурсы при достижении необходимого результата процесса;
5. Повторяемость – характеризует способность процесса создавать выходные потоки с одинаковыми характеристиками при повторных его реализациях;
6. Гибкость (адаптируемость) – способность процесса приспосабливаться к изменениям внешних условий, перестраиваться так, чтобы не снижались ни результативность, ни эффективность;
7. Стоимость – определяет совокупную стоимость выполнения функций процесса и передачи результатов от одной функции к другой. Владелец процесса - лицо, несущее полную ответственность за наделенное полномочиями в отношении этого процесса.

Вход БП (первичный или вторичный) – ресурс, необходимый для его выполнения.

Выход БП (первичный или вторичный) – результат (продукт, услуга) его выполнения.

Первичный вход	• Поток объектов, инициирующий «запуск» БП, например - заказ клиента, план закупок и т.д.
Вторичный вход	• Потоки объектов, обеспечивающие нормальное протекание БП, например – стандарты, правила, механизмы выполнения действий, оборудование и пр.
Первичный выход	• Основной результат, ради которого существует БП. • Определяется целью, назначением БП.
Вторичный выход	• Побочный продукт БП, который может быть востребован вторичными клиентами. • Не является основной целью БП.

Показатели БП – количественные и/или качественные параметры, характеризующие БП и его результат.

Показатели эффективности БП – параметры БП, характеризующие соотношение между достигнутым результатом и использованными ресурсами.

Поставщик (внешний или внутренний, первичный или вторичный) – субъект, предоставляющие ресурсы.

Внешний поставщик бизнес-процесса - организация-поставщик, подразделение или должностное лицо другой организации, предоставляющее входы (ресурсы) для выполнения бизнес-процесса компании.

Внутренний поставщик бизнес-процесса - другой бизнес-процесс компании, подразделение или должностное лицо, предоставляющее входы (ресурсы) для выполнения рассматриваемого бизнес-процесса.

Первичный поставщик бизнес-процесса - поставщик, предоставляющий для выполнения бизнес-процесса первичные входы.

Вторичный поставщик бизнес-процесса - поставщик, предоставляющий для выполнения бизнес-процесса вторичные входы.

Клиент или потребитель (внешний или внутренний, первичный или вторичный) – субъект, получающий результаты БП.

Внешний клиент (потребитель) бизнес-процесса - организация-клиент, подразделение или должностное лицо другой организации, получающее выходы (результаты) бизнес-процесса компании.

Внутренний клиент (потребитель) бизнес-процесса - другой бизнес-

процесс компании, подразделение или должностное лицо, получающее выходы (результаты) рассматриваемого бизнес-процесса.

Первичный клиент (потребитель) бизнес-процесса - клиент, получающий или использующий первичные (основные) выходы бизнес-процесса.

Бизнес-процесс предназначен для удовлетворения первичных клиентов. Вторичный клиент (потребитель) бизнес-процесса - клиент, получающий или использующий вторичные (побочные) выходы бизнес-процесса.

Операция (работа) – часть БП.

Ресурсы – информация, финансы, материалы, персонал, оборудование, инфраструктура, среда, ПО, необходимые для выполнения БП.

Функция – направление деятельности элемента организационной структуры, представляющее собой совокупность однородных операций, выполняемых на постоянной основе.

Сеть БП организации – совокупность взаимосвязанных и взаимодействующих БП, включающих все функции, выполняемые в подразделениях организации.

Классификация процессов

1. основные процессы – процессы, непосредственно связанные с созданием стоимости, ориентированные на производство товаров или оказание услуг, составляющих основную деятельность организации и обеспечивающих получение дохода.

2. вспомогательные процессы – не увеличивают ценность продукта или услуги для потребителя, но необходимы для деятельности предприятия.

3. процессы управления — это процессы, охватывающие весь комплекс функций управления на уровне каждого бизнес-процесса и системы в целом. Уровни бизнес-процессов БП верхнего уровня – это БП, получающиеся на начальных этапах процессной декомпозиции деятельности компании. К данной группе относят следующие виды деятельности: «Продажа»,

«Производство», «Закупка» и т.д.

БП нижнего уровня – это БП, получающиеся на последующих этапах процессной декомпозиции деятельности компании и являющиеся составной частью процессов верхнего уровня. К данной группе бизнес-процессов можно отнести следующие виды деятельности: «Выставление счета», «Подготовка договора», «Изготовление детали» и т.д.

Методика выполнения лабораторной работы

1) Построение бизнес-модели

Создаётся описание бизнес-процессов фирмы / организации. При этом, как правило, предполагается, что фирма / организация является узкопрофильной, то есть производит только какой-то один товар, или предоставляет только какую-то одну услугу. В реальной жизни это, как правило, не так: большинство фирм / организаций являются многопрофильными. Но при выполнении лабораторной, для упрощения ситуации, предполагается, что фирма / организация – узкопрофильная. При таком предположении задача описания бизнес-процессов существенно упрощается и становится типовой:

- выделяется основной бизнес-процесс – это тот, который приносит деньги. Он декомпозируется на составляющие бизнес-функции, то есть, выделяются действия, выполнение которых обеспечивает выполнение этого основного бизнес-процесса (см. примеры ниже, в частности рисунок 2);
- помимо основного бизнес-процесса возможно выделение поддерживающих бизнес-процессов (дополнительных к основному, обеспечивающих его выполнение). Например, для библиотеки основным бизнес-процессом будет обслуживание читателей, а поддерживающими будут бизнес-процессы «книгохранилище» и «комплектация книжного фонда». Эти поддерживающие бизнес-

процессы являются затратными, но они непосредственно связаны с основным и поддерживают его выполнение;

- поддерживающие бизнес-процессы также декомпозируются на составляющие бизнес-функции. Например, бизнес-процесс «книгохранилище» может декомпонироваться на бизнес-функции:

- поиск и выдача единиц хранения;
- приём и раскладка единиц хранения;
- отслеживание состояния единиц хранения;
- ремонт единиц хранения и др.

- почти во всех самостоятельных фирмах / организациях существуют бизнес-процессы «управление», «учёт» и «вспомогательные».

Учёт – это, обычно, бухгалтерия + формирование различного вида отчётности, выдаваемой вовне по запросам государственных или местных органов власти. Сюда же может входить функция создания рекламы.

«Вспомогательный» бизнес-процесс – это бизнес-функции «бухгалтерия», «отдел кадров», «охрана», «уборка» и другие вспомогательные операции, не имеющие прямого отношения к основному бизнес процессу.

Пример 1. ЧАСТНАЯ ТИПОГРАФИЯ

Бизнес-процессы:

1. Обслуживание клиентов – основной:
 - приём заказа,
 - подготовка макета,
 - печать образца и согласование,
 - изготовление заказа,
 - выдача заказа.
2. Ремонт оборудования – первый поддерживающий:
 - регламентные работы,
 - ремонт оборудования.
3. Снабжение – второй поддерживающий:

- определение потребностей (сбор заявок) по позициям номенклатуры,
 - заказ/закупка бумаги, картриджей, типографской краски и др.,
 - заказ/закупка запасных частей и материалов,
 - заказ/закупка оборудования и инструментов,
 - спецодежда, инвентарь, оргтехника, канцпринадлежности.
4. Учёт и бухгалтерия:
- учёт заказов, калькуляция,
 - учёт материалов,
 - бухгалтерская отчётность,
 - другая отчётность (налог., пенс., соцстрах, госстат., местная и
 -
 -
 - расчёт зарплаты.
5. Вспомогательные бизнес-функции:
- отдел кадров,
 - охрана,
 - уборка.
6. Управление.

Основной бизнес-процесс в данной фирме – «обслуживание клиентов»: печать по заказам рекламных материалов, календарей, визиток, брошюр и другой графической печатной продукции, не требующей переплёта.

Поддерживающие бизнес-процессы: «снабжение» и «ремонт и обслуживание техники». Они являются затратными и непосредственно обеспечивают выполнение основного бизнес-процесса.

Примечание: обратите внимание – бухгалтерия может находиться в бизнес- процессах «учёт» или «вспомогательные». Где конкретно – определяется спецификой конкретного предприятия/организации, то есть, степенью привязанности бухгалтерии к основному бизнес-процессу. Например, в приведённом выше примере, бухгалтерия включена в бизнес- процесс «учёт и бухгалтерия», по-скольку она явно обслуживает основной бизнес-процесс: помимо чисто бухгалтерских функций она обеспечивает также калькуляцию.

Состав бизнес-функций и матрица ответственности

Даже если информационная система предназначена для автоматизации маленькой фирмы, для более качественного проектирования следует предположить, что объём работ, выполняемых фирмой, требует как минимум 20-30 человек штата. Тогда будут видны (станут заметны) те работы, которые в маленькой фирме со штатом 3-5 человек не видны: их выполняют «по совместительству» (неявно) сотрудники или хозяин фирмы, иногда даже не замечая. В средней же фирме эти работы приходится выделять официально в отдельные бизнес-функции и поручать их выполнение отдельным штатным единицам (см. данные к примеру 1 (частная типография), приведенные в табл. 1, 2).

Таблица 1 - Состав бизнес-процессов: функции и работы(частная типография)

Бизнес-процесс	Функции	Работы	Подразделение	Должности
1. Управление	1.1. Управление	1.1. Управление фирмой	1. Управление	(1.1.) 1. Владелец типографии (хозяин). (1.1.) 2. Директор. (1.1.) 3. Исполнительный директор
2. Основной: обслуживание клиентов	2.1. Приём заказов.	2.1. Приём заказов.	2. Производственный отдел	(2.1.) 1. Владелец типографии (хозяин). (2.1.) 2. Менеджер
	2.2. Подготовка макета.	2.2. Дизайн. Обработка графики.		(2.2., 2.3., 2.4.) 1. Дизайнер.
	2.3. Печать образцов. Печать макета на	2.3. Печать на принтере образцов. Печать на принтере макета		

Структура ПО

Далее по матрице ответственности определяется состав рабочих мест фирмы / организации и принимается решение о том, какие из них будут автоматизироваться, то есть определяется список АРМ – автоматизированных рабочих мест, тех рабочих мест, которые будут оснащаться компьютерами и соответствующим ПО разрабатываемой информационной системы. Этот список должен формироваться с учётом требований технического задания на информационную систему, поскольку именно в ТЗ оговаривается, что собственно должно автоматизироваться.

Таблица 4

Список АРМ

Автоматизируемые рабочие места:	Количество АРМ
АРМ «Управление»	1
АРМ «Дизайнер»	3
АРМ «Снабженец»	1
АРМ «Бухгалтер»	1
АРМ «Ремонт»	1
Возможно, АРМ Менеджера по приёму заказов	1

Определяем взаимосвязи этих автоматизируемых рабочих мест – строим укрупнённую (обобщённую) структуру информационной системы (см. рис. 2). По этой структуре уже видно, какое программное обеспечение (с какой функциональностью) для каждого автоматизируемого рабочего места нужно создавать. Этот документ (рис. 2) является основой для дальнейшей разработки информационной системы. На основе данной структуры далее разрабатываются частные технические задания (ЧТЗ) на компоненты информационной системы: АРМ, программные комплексы, протоколы, интерфейсы и отдельные .

Задание:

1. Разработать модель бизнес-процессов обследуемого предприятия / организации / фирмы (заказчика), для которой разрабатывается вариант информационной системы. Определить основные, дополнительные, вспомогательные бизнес-процессы, а также бизнес-процесс управления.

2. Определить состав бизнес-функций по каждому бизнес-процессу. Описать работы, выполняемые в рамках каждой бизнес-функции.

3. Определить штат сотрудников для выполнения описанного в пункте 2 состава бизнес-функций. Описать: кто, на каком рабочем месте выполняет перечисленные в пункте 2 работы. Построить матрицу ответственности.

4. Построить структуру программного обеспечения проектируемой информационной системы. Для этого необходимо изучить работу с онлайн программой, моделирующей бизнес-процессы, которую можно открыть по ссылке <https://www.bpsimulator.com/>.

5. Изучить принцип работы с программой моделирования бизнес-процессов.

6. Провести моделирование бизнес-процессов своей информационной системы.

Порядок сдачи лабораторной работы: Представить отчёт, содержащий модель бизнес-процессов предприятия / организации / фирмы (поиндивидуальному варианту) для разработки Информационной системы.

Отчет должен содержать следующую информацию:

- описание процесса построения бизнес-модели и представлении модели бизнес-процессов на рисунке;
- состав бизнес-функций (и выполняемых работ по ней) по каждому бизнес-процессу (в виде таблицы);

- матрица ответственности:
 - сверху – бизнес-функции / работы;
 - слева – подразделения и сотрудники;
 - на пересечении (в клеточках матрицы) – рабочие места, на которых выполняются соответствующие функции / работы;
- структура программного обеспечения проектируемой информационной системы: модули рабочих мест и их взаимосвязи (рисунок).

Практическая работа 4.

IDEF()-диаграммы. Методы оценки бизнес-процессов

Цель: изучить построение функциональных диаграмм и их применение для оценки бизнес-процессов, а также стандарты IDEF().

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

Методология IDEF0 является следующим этапом развития хорошо известного графического языка описания функциональных систем SADT. Термин SADT

(Structured Analysis and Design Technique) обозначает технологию структурного анализа и проектирования. Термин IDEF обозначает Icam DEFinition - нотацию ICAM, которая является основной частью программы ICAM (Integrated Computer-Aided Manufacturing - интегрированная компьютеризация производства), которая проводилась по инициативе BBC США. Методологии функционального моделирования семейства IDEF стандартизированы. С их помощью можно эффективно отображать и анализировать модели деятельности широкого спектра сложных систем.

Основной принцип методологии IDEF – это представление любой изучаемой системы в виде набора взаимодействующих и взаимосвязанных блоков, отображающих процессы, операции, действия, происходящие в изучаемой системе. В IDEF0 все, что происходит в системе и ее элементах, называется функциями. Каждой функции ставится в соответствие блок. На диаграмме IDEF0 блок представляет собой прямоугольник. Интерфейсы, посредством которых блок взаимодействует с другими блоками или с внешней по отношению к моделируемой системе средой, представляются стрелками, входящими в блок или выходящими из него. Входящие стрелки показывают, какие условия должны быть одновременно выполнены, чтобы функция, описываемая блоком, осуществилась. Диаграммы IDEF0 основаны на простой

графике блоков и стрелок, метки для описания блоков и стрелок выполняются на естественном языке. Последовательная декомпозиция диаграмм строится по иерархическому принципу, при котором на верхнем уровне отображаются основные функции, а затем происходит их детализация и уточнение.

Разработка модели в IDEF0 представляет собой пошаговую, итеративную процедуру. На каждом шаге итерации разработчик предлагает вариант модели, который подвергают обсуждению, рецензированию и последующему редактированию, после чего цикл повторяется.

Основные определения (понятия) методологии и языка IDEF0.

1. Блок: прямоугольник, содержащий имя и номер и используемый для описания функции.
2. Ветвление: разделение стрелки на два или большее число сегментов.
3. Внутренняя стрелка: входная, управляющая или выходная стрелка, концы которой связывают источник и потребителя, являющиеся блоками одной диаграммы. Отличается от граничной стрелки.
4. Входная стрелка: класс стрелок, которые отображают вход IDEF0-блока, то есть данные или материальные объекты, которые преобразуются функцией в выход. Входные стрелки связываются с левой стороной блока IDEF0.
5. Выходная стрелка: класс стрелок, которые отображают выход IDEF0-блока, то есть данные или материальные объекты, произведенные функцией. Выходные стрелки связываются с правой стороной блока IDEF0.
6. Глоссарий: список определений для ключевых слов, фраз и аббревиатур, связанных с узлами, блоками, стрелками или с моделью IDEF0 в целом.
7. Граничная стрелка: стрелка, один из концов которой связан с источником или потребителем, а другой не присоединен ни к какому блоку на диаграмме. Отображает связь диаграммы с другими блоками системы и отличается от внутренней стрелки.
8. Декомпозиция: разделение моделируемой функции на функции – компоненты.

9. Дерево узлов: представление отношений между родительскими и дочерними узлами модели IDEF0 в форме древовидного графа.

10. Диаграмма A-0: специальный вид (контекстной) диаграммы IDEF0, состоящей из одного блока, описывающего функцию верхнего уровня, ее входы, выходы, управления, и механизмы, вместе с формулировками цели модели и точки зрения, с которой строится модель.

11. Диаграмма: часть модели, описывающая декомпозицию блока.

12. Диаграмма-иллюстрация (FEO): графическое описание, используемое, для сообщения специфических фактов о диаграмме IDEF0. При построении диаграмм FEO можно не придерживаться правила IDEF0.

13. Дочерний блок: блок на дочерней (порожденной) диаграмме.

14. Дочерняя диаграмма: диаграмма, детализирующая родительский (порождающий) блок.

15. Имя блока: глагол или глагольный оборот, помещенный внутри блока и описывающий моделируемую функцию.

16. Интерфейс: разделяющая граница, через которую проходят данные или материальные объекты; соединение между двумя или большим числом компонентов модели, передающее данные или материальные объекты от одного компонента к другому.

17. Код ICOM: аббревиатура (Input - Вход, Control - Управление, Output - Выход, Mechanism – Механизм), код, обеспечивающий соответствие граничных стрелок дочерней диаграммы со стрелками родительского блока; используется для ссылок.

18. Контекст: окружающая среда, в которой действует функция (или комплект функций на диаграмме).

19. Контекстная диаграмма: диаграмма, имеющая узловой номер A-n ($n \geq 0$), которая представляет контекст модели, Диаграмма A-0, состоящая из одного блока, является необходимой (обязательной) контекстной диаграммой; диаграммы с узловыми номерами A-1, A-2,... - дополнительные контекстные диаграммы.

20. Метка стрелки: существительное или оборот существительного,

связанные со стрелкой или сегментом стрелки и определяющие их значение.

21. Модель IDEF0: графическое описание системы, разработанное с определенной целью и с выбранной точки зрения. Комплект одной или более диаграмм IDEF0, которые изображают функции системы с помощью графики, текста и глоссария.

22. Номер блока: число (0 – 6), помещаемое в правом нижнем углу блока и однозначно идентифицирующее блок на диаграмме.

23. Перечень узлов: список, часто ступенчатый, показывающий узлы модели IDEF0 в упорядоченном виде. Имеет то же значение и содержание, что и дерево узлов (п. 9).

24. Примечание к модели: текстовый комментарий, являющийся частью диаграммы IDEF0 и используемый для записи факта, не нашедшего графического изображения.

25. Родительская диаграмма: диаграмма, которая содержит родительский блок.

26. Родительский блок: блок, который подробно описывается дочерней диаграммой.

27. Связывание/развязывание: объединение значений стрелок в составное значение (связывание в «пучок»), или разделение значений стрелок (развязывание «пучка»), выраженные синтаксисом слияния или ветвления стрелок.

28. Сегмент стрелки: сегмент линии, который начинается или заканчивается на стороне блока, в точке ветвления или слияния, или на границе (несвязанный конец стрелки).

29. Семантика: значение синтаксических компонентов языка.

30. Синтаксис: Структурные компоненты или характеристики языка и правила, которые определяют отношения между ними.

31. Слияние: объединение двух или большего числа сегментов стрелок в один сегмент.

32. С-номер: номер, создаваемый в хронологическом порядке и используемый для идентификации диаграммы и прослеживания ее истории;

может быть

33. использован в качестве ссылочного выражения при определении конкретной версии диаграммы.

34. Стрелка: направленная линия, состоящая из одного или нескольких сегментов, которая моделирует открытый канал или канал, передающий данные или материальные объекты от источника (начальная точка стрелки), к потребителю (конечная точка с «наконечником»). Имеется 4 класса стрелок: входная стрелка, выходная стрелка, управляющая стрелка, стрелка механизма (включает стрелку вызова).

35. Стрелка вызова: вид стрелки механизма, который обозначает обращение из блока данной модели (или части модели) к блоку другой модели (или другой части той же модели) и обеспечивает связь между моделями или между разными частями одной модели.

36. Стрелка механизма: класс стрелок, которые отображают механизмы IDEF0, то есть средства, используемые для выполнения функции; включает специальный случай стрелки вызова. Стрелки механизмов связываются с нижней стороной блока IDEF0.

37. Стрелка, помещенная в туннель (туннельная стрелка): стрелка (со специальной нотацией), не удовлетворяющая обычному требованию, согласно которому каждая стрелка на дочерней диаграмме должна соответствовать стрелкам на родительской диаграмме.

38. Текст: любой текстовый (не графический) комментарий к графической диаграмме IDEF0.

39. Тильда: небольшая ломаная (волнистая) линия, используемая для соединения метки с конкретным сегментом стрелки или примечания модели с компонентом диаграммы.

40. Точка зрения: указание на должностное лицо или подразделение организации, с позиции которого разрабатывается модель.

41. Узел: блок, порождающий дочерние блоки; родительский блок.

42. Узловая ссылка: код, присвоенный диаграмме, для ее идентификации и определения положения в иерархии модели; формируется из сокращенного

имени модели и узлового номера диаграммы с дополнительными расширениями.

43. Узловой номер диаграммы: часть узловой ссылки диаграммы, которая соответствует номеру родительского блока.

44. Узловой номер: код, присвоенный блоку и определяющий его положение в иерархии модели; может быть использован в качестве подробного ссылочного выражения.

45. Управляющая стрелка: класс стрелок, которые в IDEF0 отображают управления, то есть условия, при выполнении которых выход блока будет правильным. Данные или объекты, моделируемые как управления, могут преобразовываться функцией, создающей соответствующий выход. Управляющие стрелки связываются с верхней стороной блока IDEF0.

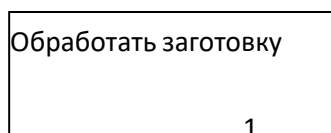
46. Функция: деятельность, процесс или преобразование (моделируемые блоком IDEF0), идентифицируемое глаголом или глагольной формой, которая описывает, что должно быть выполнено.

47. Цель: краткая формулировка причины создания модели.

В основе методологии IDEF0 лежат четыре основных понятия

- функциональный блок (Activity Box);
- интерфейсная дуга (Arrow) или стрелка;
- декомпозиция (Decomposition);
- глоссарий (Glossary).

Функциональный блок показан на рисунке 1.1. Внутри каждого блока размещается его имя и номер. Имя должно быть активным глаголом или



глагольным оборотом, описывающим функцию. Номер блока размещается в правом нижнем углу. Номера блоков используются для их идентификации на диаграмме и в соответствующем тексте.

Рисунок 1.1 – Блок

Стрелка (дуга) формируется из одного или более отрезков прямых линий и наконечника на одном конце. Отрезки стрелок могут быть прямыми или ломаными; в последнем случае горизонтальные и вертикальные отрезки стрелки

сопрягаются дугами, имеющими угол 90° . Правила построения стрелок показаны на рисунке 1.2.

Концы стрелок должны касаться внешней границы функционального блока, но не должны пересекать ее. Стрелки должны присоединяться к блоку на его сторонах, а не в углах.

Каждая сторона функционального блока имеет стандартное значение и однозначно определяет роль присоединенной стрелки. Стрелки, входящие в левую сторону блока обозначают входы. Входы преобразуются или расходуются функцией, чтобы создать то, что появится на ее выходе. Стрелки, входящие в блок сверху, соответствуют управлению. Стрелки, выходящие справа, обозначают выходы, т.е. данные или материальные объекты, произведенные функцией.

Стрелки, подключенные к нижней стороне блока, представляют механизмы. Стрелки, направленные вверх, идентифицируют средства, поддерживающие выполнение функции. Другие средства могут наследоваться из родительского блока. Стрелки механизма, направленные вниз, являются стрелками вызова. Стрелки вызова обозначают обращение из данной модели или из данной части модели к блоку, входящему в состав другой модели или другой части модели, обеспечивая их связь, т.е. разные модели или разные части одной и той же модели могут совместно использовать один и тот же элемент (блок).

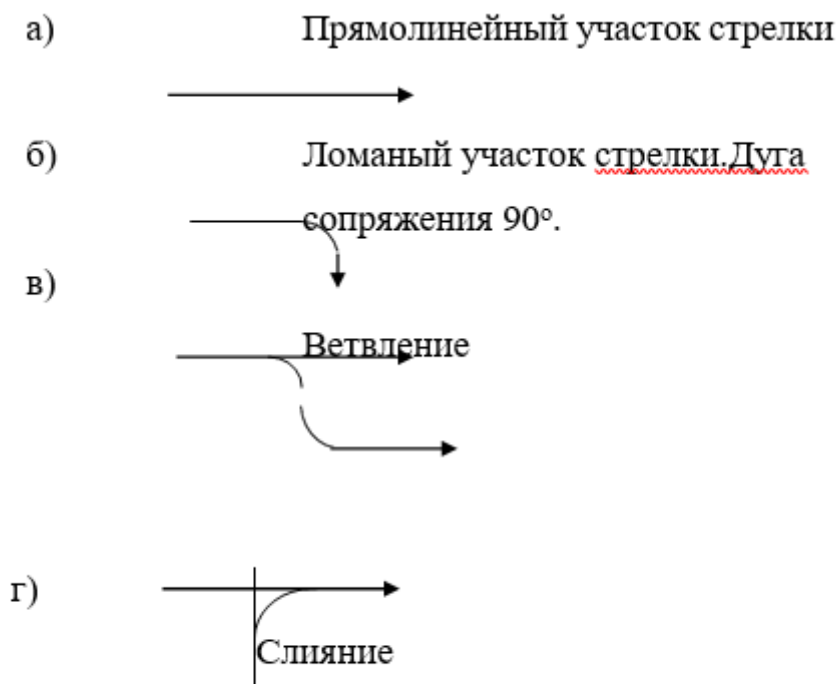


Рисунок 1.2 – Построение стрелок

Стандартное расположение стрелок показано на рисунке 1.3.

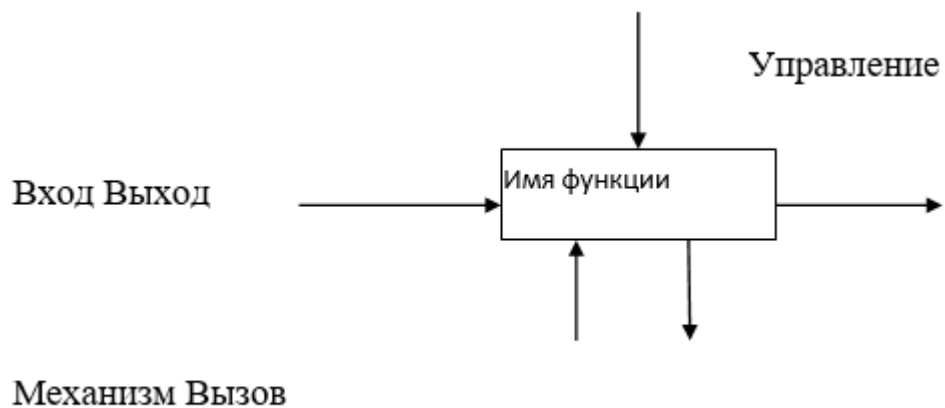


Рисунок 1.3 – Стандартное расположение стрелок

Примеры имен функций (функциональных блоков):

- производить детали;
- планировать ресурсы;
- наблюдать за выполнением;
- проектировать систему;
- эксплуатировать;
- разработать чертежи;
- изготовить деталь;
- проверять деталь.

Стрелки идентифицируют данные или материальные объекты, необходимые для выполнения функции или производимые ею. Каждая стрелка должна быть помечена существительным или оборотом существительного, например:

- спецификации;
- отчет об испытаниях;
- бюджет;
- конструкторские требования;
- конструкция детали;
- директива;
- инженер-конструктор
- плата в сборе;

– требования.

В метках стрелок не должны использоваться такие общеупотребительные термины, как: функция, вход, управление, выход, механизм, вызов.

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком с граничными стрелками. Эта диаграмма называется А-0. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой. Поскольку единственный блок представляет весь объект, его имя – общее для всего проекта. Это же справедливо и для всех стрелок диаграммы, поскольку они представляют полный комплект внешних интерфейсов объекта. Диаграмма А-0 устанавливает область моделирования и ее границу. Пример диаграммы А-0 показан на рисунке 1.4.

Контекстная диаграмма А-0 также должна содержать краткие утверждения, определяющие точку зрения должностного лица или подразделения, с позиций которого создается модель, и цель, для достижения которой ее разрабатывают. Изменение точки зрения, приводит к рассмотрению других аспектов объекта. Аспекты, важные с одной точки зрения, могут не появиться в модели, разрабатываемой с другой точки зрения на тот же самый объект. Формулировка цели выражает причину создания модели, т.е. содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет ее структуру.

Наиболее важные свойства объекта обычно выявляются на верхних уровнях иерархии. По мере декомпозиции функции (функционального блока) верхнего уровня и разбиения на подфункции, эти свойства уточняются. Каждая подфункция может быть представлена декомпозицией на элементы следующего уровня, и так происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования. Каждая подфункция моделируется отдельным блоком. Каждый родительский блок подробно описывается дочерней диаграммой на более низком уровне. Все дочерние диаграммы не должны выходить за пределы области контекстной диаграммы верхнего уровня.

Дочерняя диаграмма

Функция, представленная на контекстной диаграмме верхнего уровня, может быть разложена на основные подфункции путем создания дочерней диаграммы. В свою очередь, каждая из этих подфункций может быть разложена на составные части путем создания дочерней диаграммы следующего, более низкого уровня. Каждая дочерняя диаграмма содержит дочерние блоки и стрелки, обеспечивающие дополнительную детализацию родительского блока.

Дочерняя диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский блок, но описывает ее более подробно. Поэтому дочернюю диаграмму можно рассматривать как вложенную в свой родительский блок. Эта структура показана на рисунке 1.5.

Родительская диаграмма

Родительской называется диаграмма, которая содержит один или более родительских блоков. Каждая не контекстная диаграмма является также дочерней диаграммой, поскольку, по определению, она подробно описывает некоторый родительский блок. Таким образом, любая диаграмма может быть как родительской диаграммой (содержать родительские блоки), так и дочерней (подробно описывать собственный родительский блок). Аналогично, блок может быть как родительским (подробно описываться дочерней диаграммой) так и дочерним (появляющимся на дочерней диаграмме).

В методологии IDEF0 существует 6 (шесть) типов отношений между блоками в пределах одной диаграммы:

- доминирование;
- управление;
- выход - вход;
- обратная связь по управлению;
- обратная связь по входу;
- выход – механизм

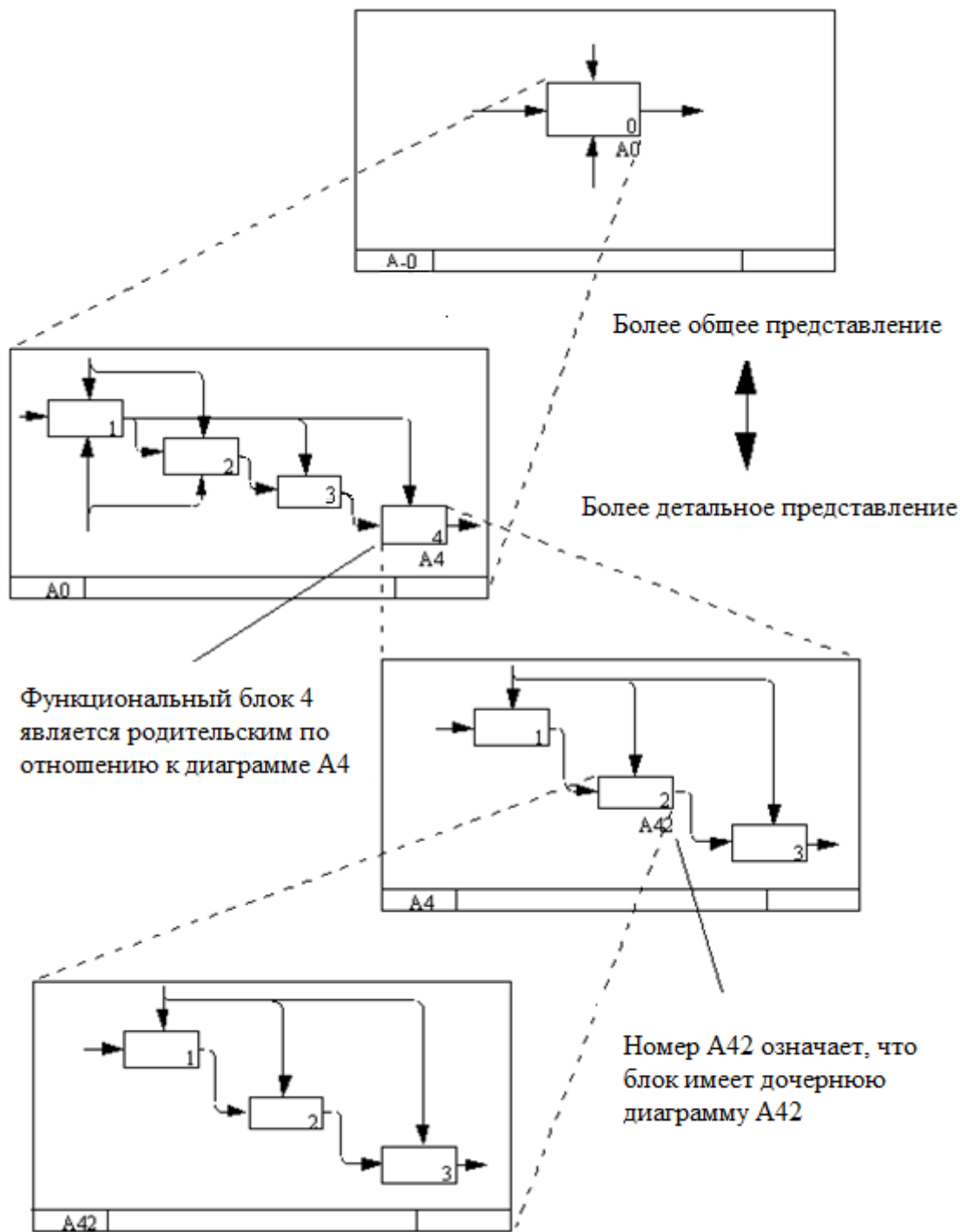


Рисунок 1.5 - Декомпозиция функциональных блоков

Доминирование определяется взаимным расположением блоков на диаграмме. Предполагается, что блоки, расположенные на диаграмме выше и левее, доминируют над блоками, расположенными ниже и правее. Доминирование подразумевает влияние, которое один блок оказывает на другие блоки диаграммы. Остальные пять отношений описывают связи между блоками и изображаются соответствующими стрелками.

Стрелки, помещенные в «туннель»

Туннель - круглые скобки в начале и/или окончании стрелки. Туннельные стрелки означают, что данные, выраженные этими стрелками, не рассматриваются на родительской диаграмме и/или на дочерней диаграмме. Стрелка, помещенная в туннель там, где она присоединяется к блоку, означает, что данные, выраженные этой стрелкой, не обязательны на следующем уровне декомпозиции. Стрелка, помещаемая в туннель на свободном конце, означает, что выраженные ею данные отсутствуют на родительской диаграмме. Туннель используется, чтобы не загромождать диаграмму.

Правила построения диаграмм

1. В составе модели должна присутствовать контекстная диаграмма A-0, которая содержит только один блок.

2. Блоки на диаграмме должны располагаться по диагонали – от левого верхнего угла диаграммы до правого нижнего в порядке присвоенных номеров. Блоки на диаграмме, расположенные вверху слева доминируют над блоками, расположенными внизу справа. Расположение блоков на листе диаграммы отражает авторское понимание доминирования. Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные.

3. Не контекстные диаграммы должны содержать не менее трех и не более шести блоков. Эти ограничения поддерживают сложность диаграмм на уровне, доступном для чтения, понимания и использования.

4. Каждый блок не контекстной диаграммы получает номер, помещаемый в правом нижнем углу; порядок нумерации - от верхнего левого к нижнему правому блоку (номера от 1 до 6).

5. Каждый блок, подвергнутый декомпозиции, должен иметь ссылку на дочернюю диаграмму; ссылка (например, узловой номер, C-номер или номер страницы) помещается под правым нижним углом блока.

6. Имена блоков (выполняемых функций) и метки стрелок должны быть уникальными. Если метки стрелок совпадают, это значит, что стрелки отображают тождественные данные.

7. При наличии стрелок со сложной топологией целесообразно повторить метку для удобства ее идентификации.

8. Следует обеспечить максимальное расстояние между блоками и поворотами стрелок, а также между блоками и пересечениями стрелок для облегчения чтения диаграммы. Одновременно уменьшается вероятность перепутать две разные стрелки.

9. Блоки всегда должны иметь хотя бы одну управляющую и одну выходную стрелку, но могут не иметь входных стрелок.

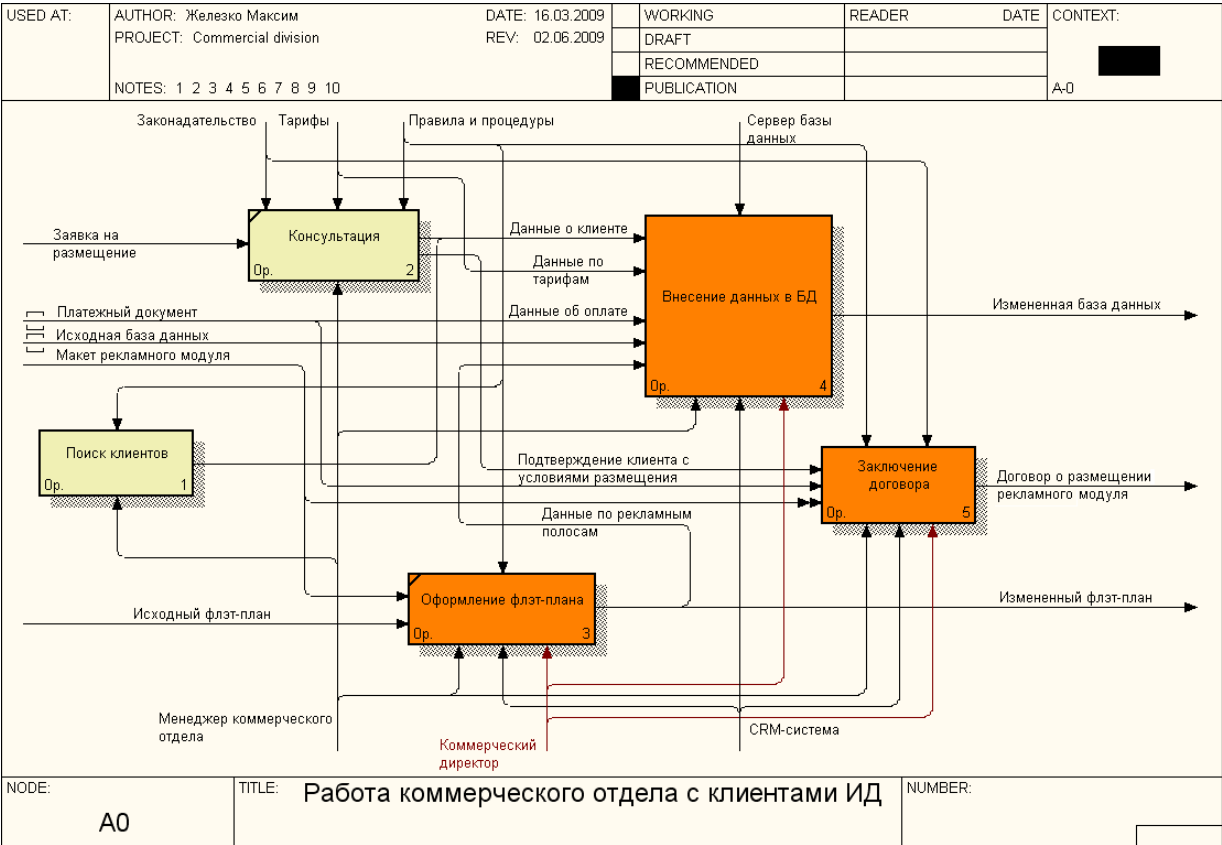
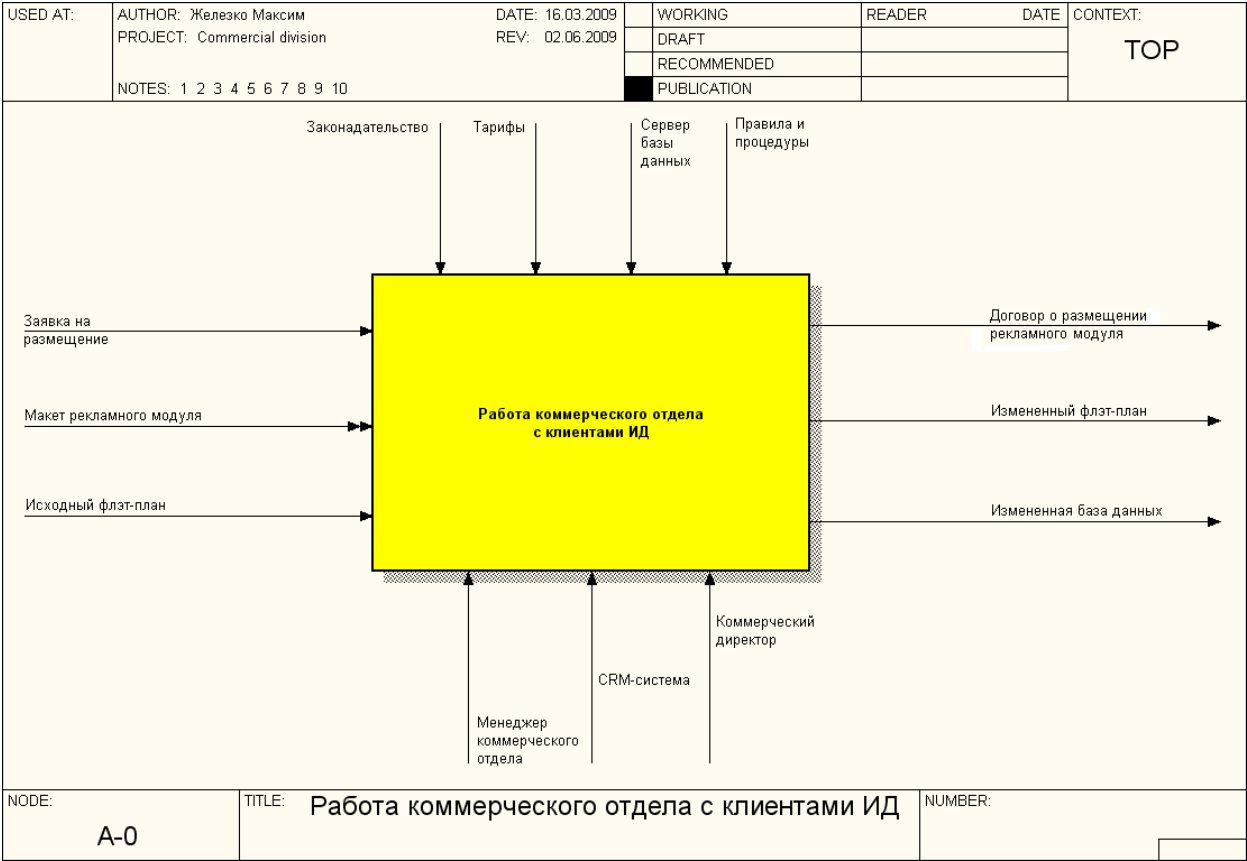
10. Если одни и те же данные служат и для управления, и для входа, вычерчивается только стрелка управления. Этим подчеркивается управляющий характер данных и уменьшается сложность диаграммы.

11. Максимально увеличенное расстояние между параллельными стрелками облегчает размещения меток, их чтение и позволяет проследить пути стрелок.

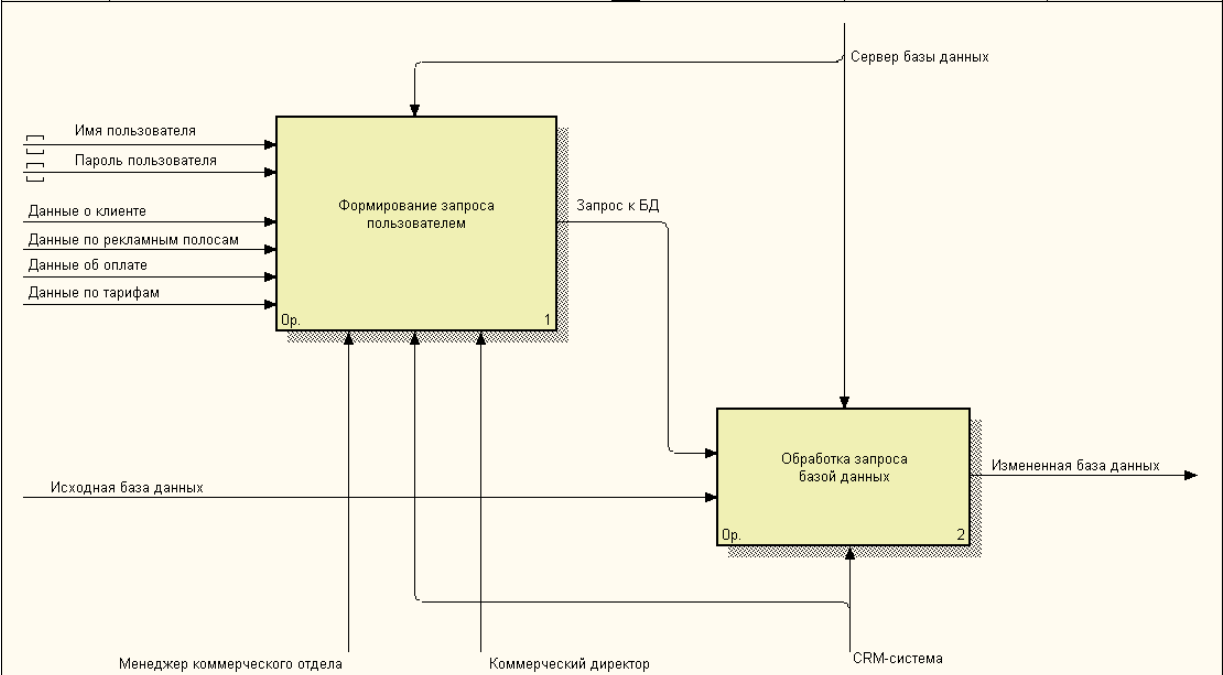
12. Стрелки связываются (сливаются), если они представляют сходные данные и их источник не указан на диаграмме.

13. Обратные связи по управлению должны быть показаны стрелками, расположенными сверху и над блоком. Обратные связи по входу должны быть показаны стрелками, расположенными внизу и под блоком. Так же показываются обратные связи посредством механизма. Таким образом, обеспечивается показ обратной связи при минимальном числе линий и пересечений.

Примеры

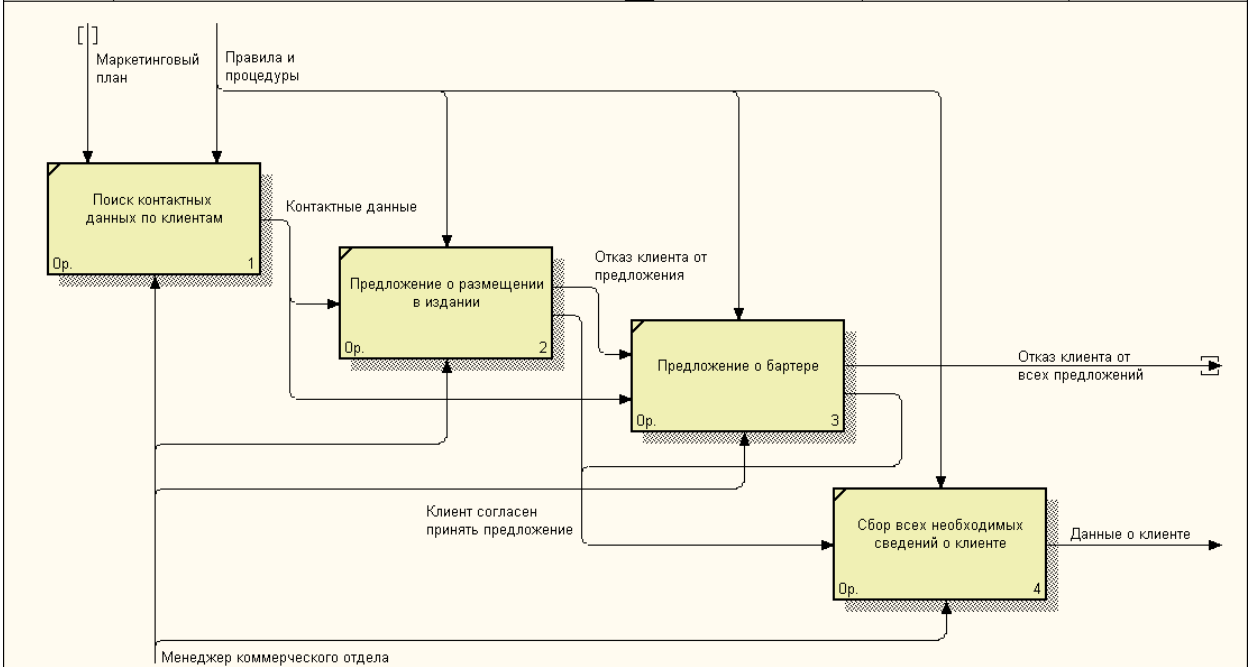


USED AT:	AUTHOR: Железко Максим PROJECT: Commercial division NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 17.03.2009 REV: 02.06.2009	WORKING	READER	DATE	CONTEXT: A0
			DRAFT			
			RECOMMENDED			
			PUBLICATION			

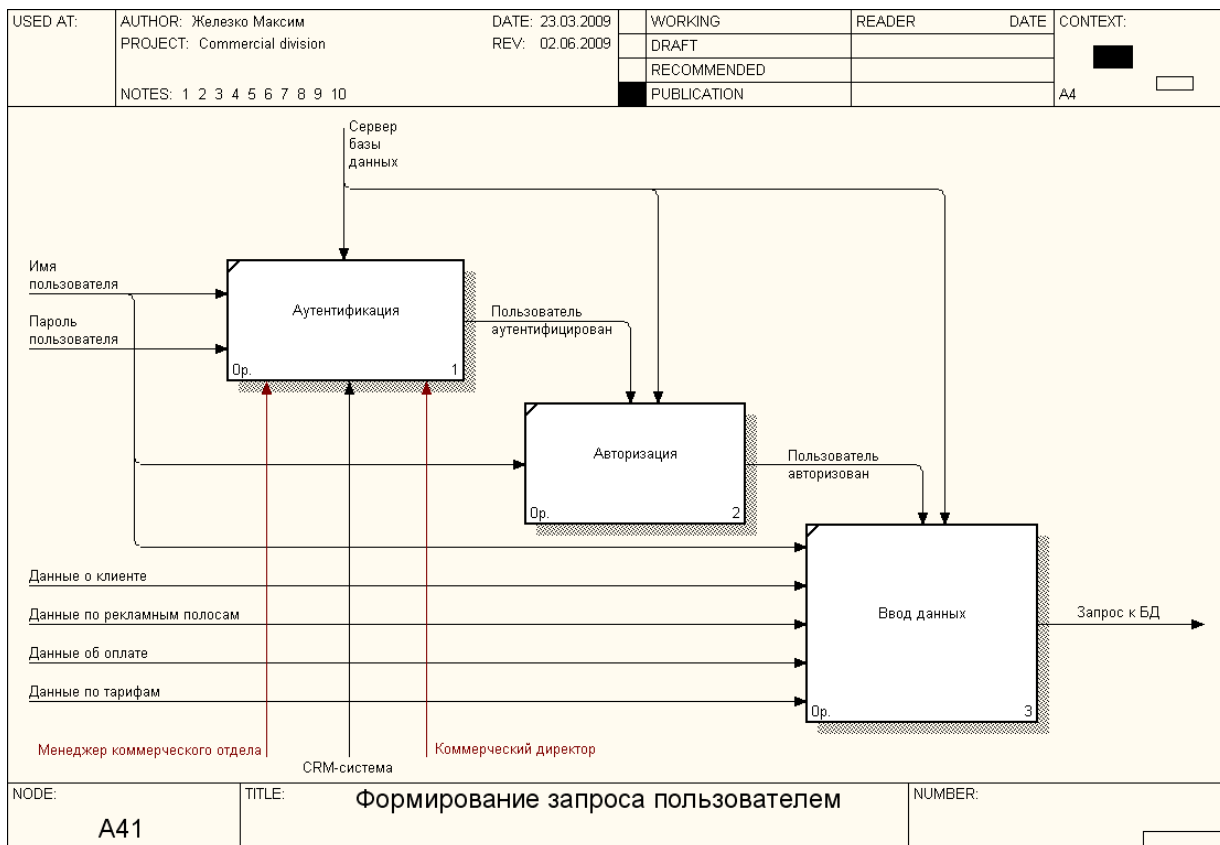
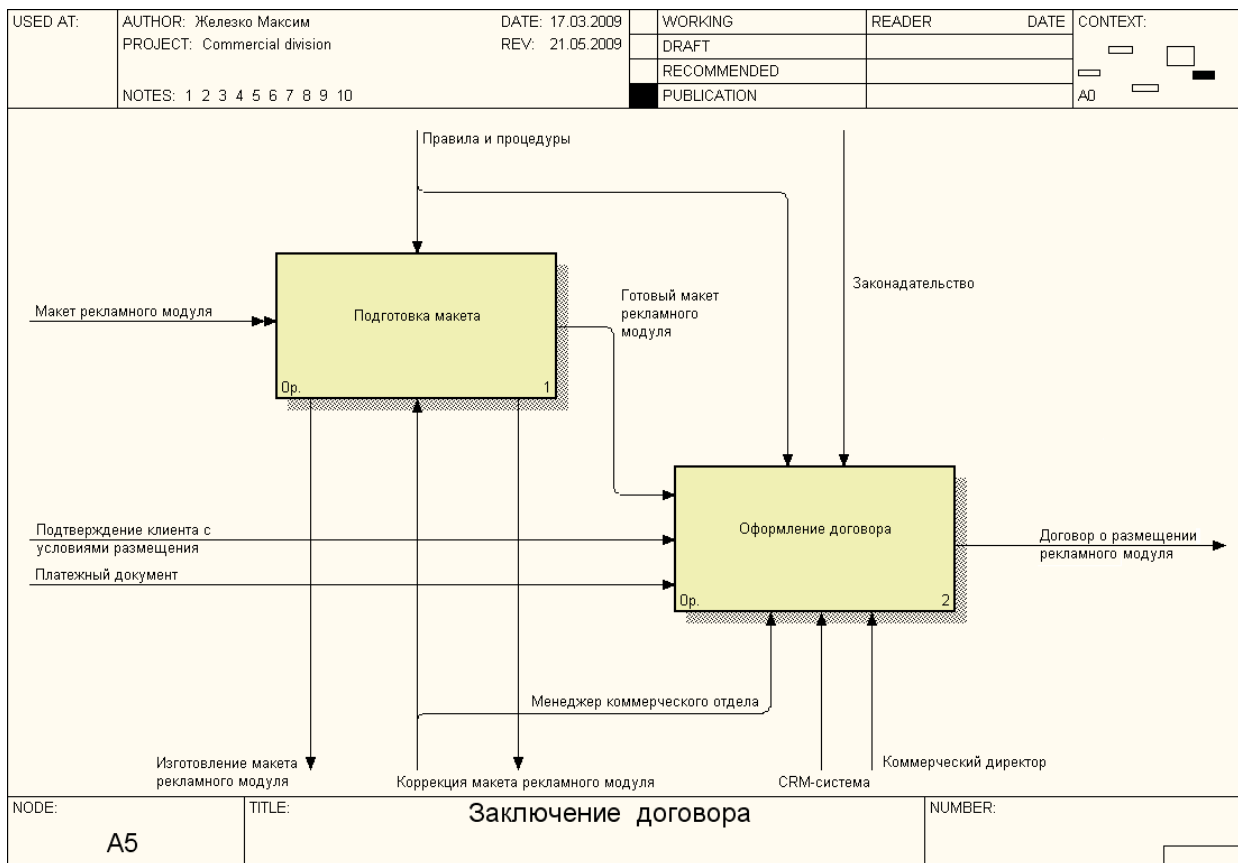


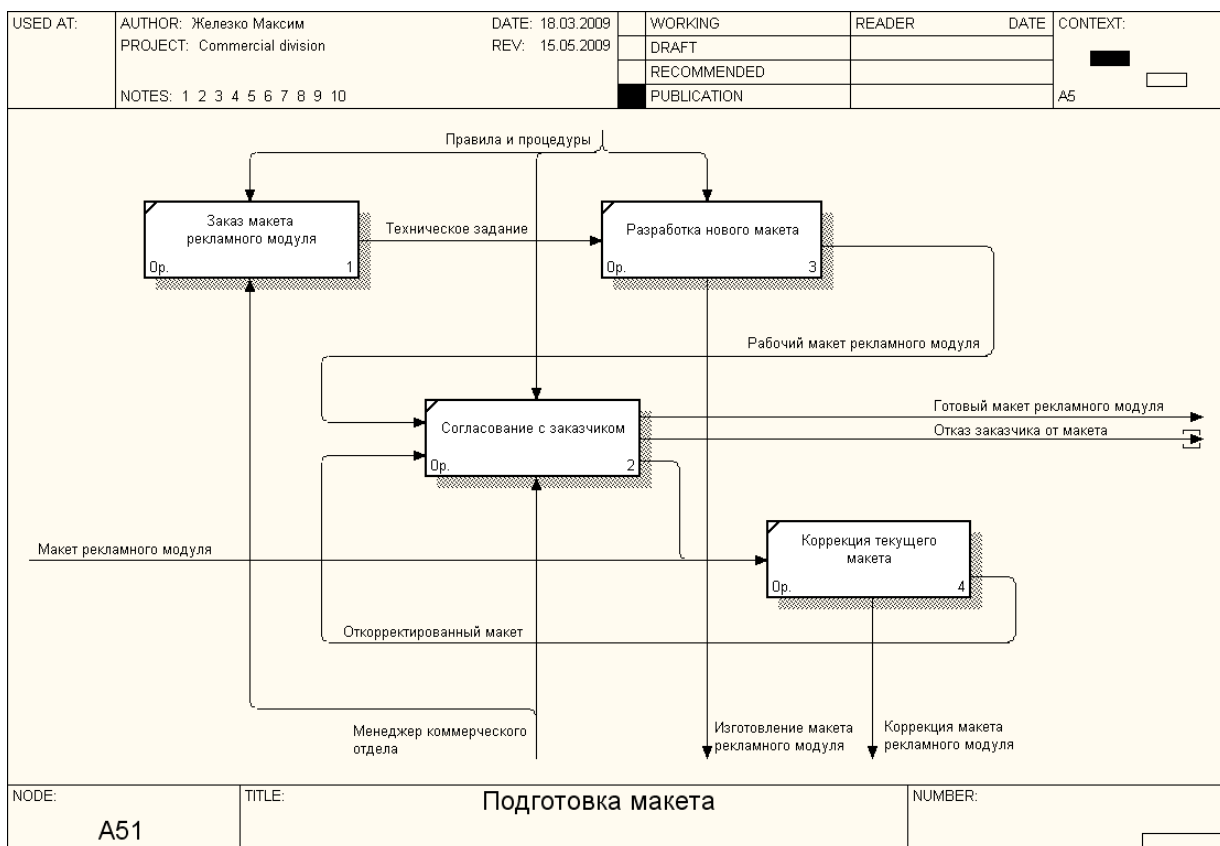
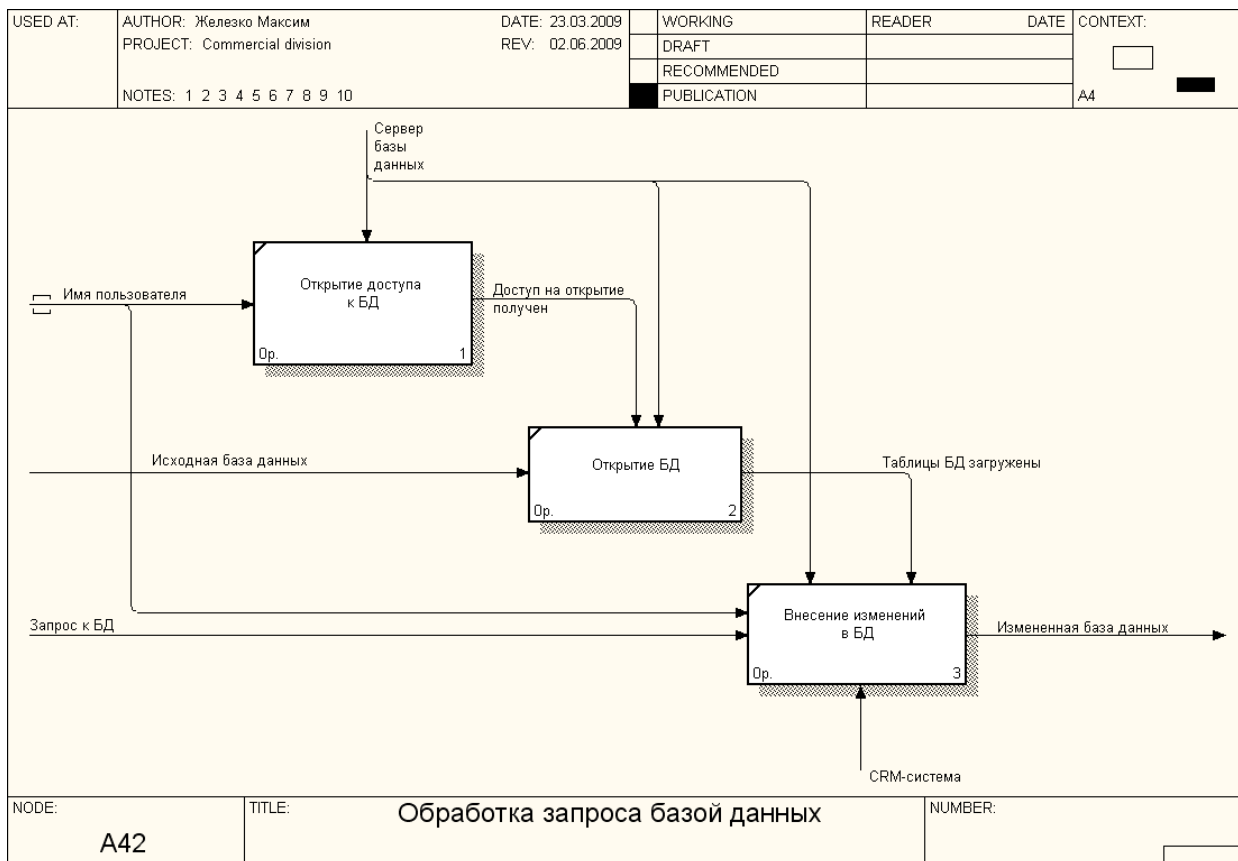
NODE:	TITLE:	NUMBER:
A4	Внесение данных в БД	

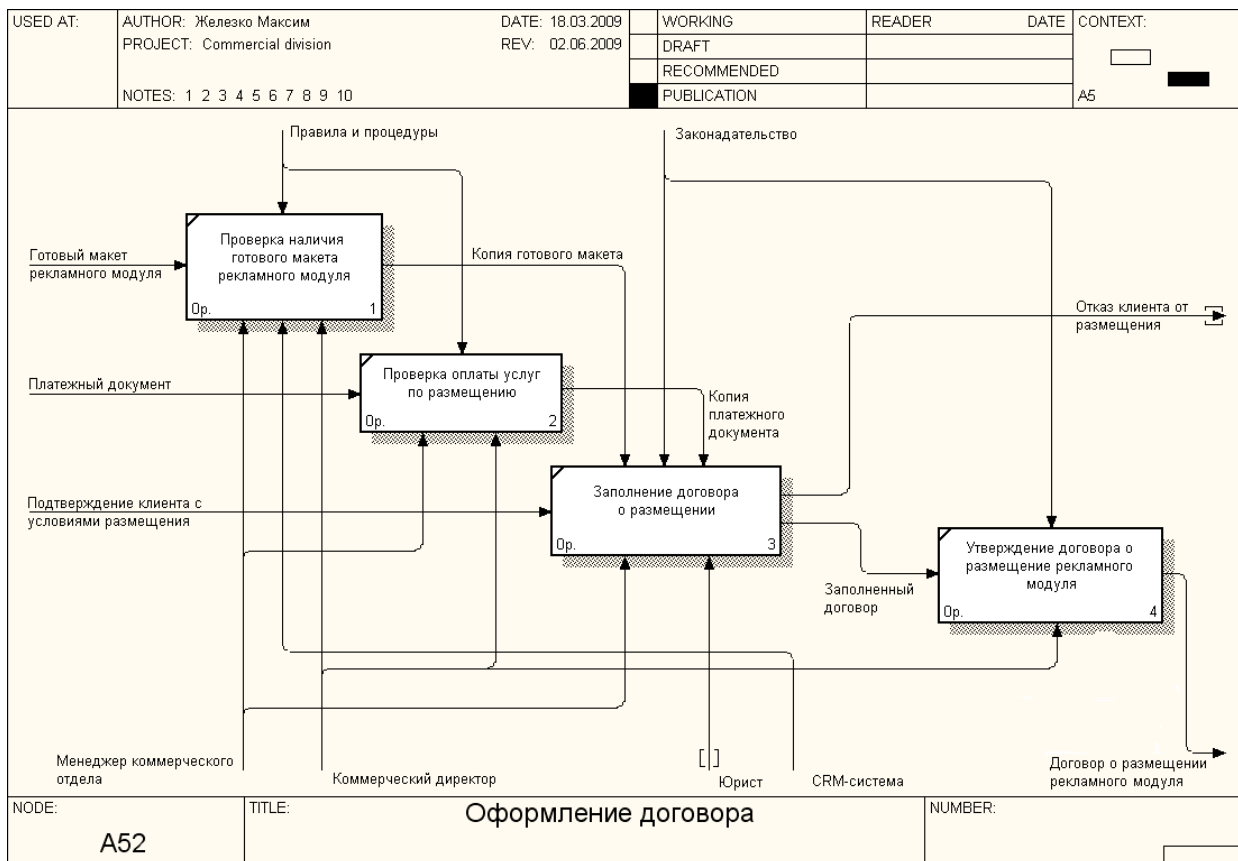
USED AT:	AUTHOR: Железко Максим PROJECT: Commercial division NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 23.03.2009 REV: 02.06.2009	WORKING	READER	DATE	CONTEXT: A0
			DRAFT			
			RECOMMENDED			
			PUBLICATION			



NODE:	TITLE:	NUMBER:
A1	Поиск клиентов	







Контрольные вопросы

1. Изучите и проанализируйте возможности функциональных диаграмм для оценки бизнес-процессов.
2. Перечислите основные компоненты функциональной диаграммы.
3. Какие диаграммы называются контекстными?
4. Какие диаграммы называются родительскими?
5. Какие диаграммы называются дочерними?
6. Каким образом на функциональных диаграммах отображается доминирование одних функциональных блоков над другими?

Практическая работа № 5.

Знакомство с работой Git

Цель работы: ознакомиться с основами работы с **Git**

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

Разработка – это почти всегда командная игра. Умение работать в команде – одно из требований к разработчикам.

Чтобы эффективно работать в команде, мало знать синтаксис языка, ключевые библиотеки и уметь обращаться с базами данных. Необходимо уметь работать в удобной для команды системе контроля версий.

О системах контроля версий их преимуществах и недостатках можно почитать:

https://ru.hexlet.io/courses/git_base/lessons/vcs_intro/theory_unit
<https://devpractice.ru/git-for-beginners-part-1-what-is-vcs/>
<https://biz30.timedoctor.com/ru/система-контроля-версий/>

и пр.

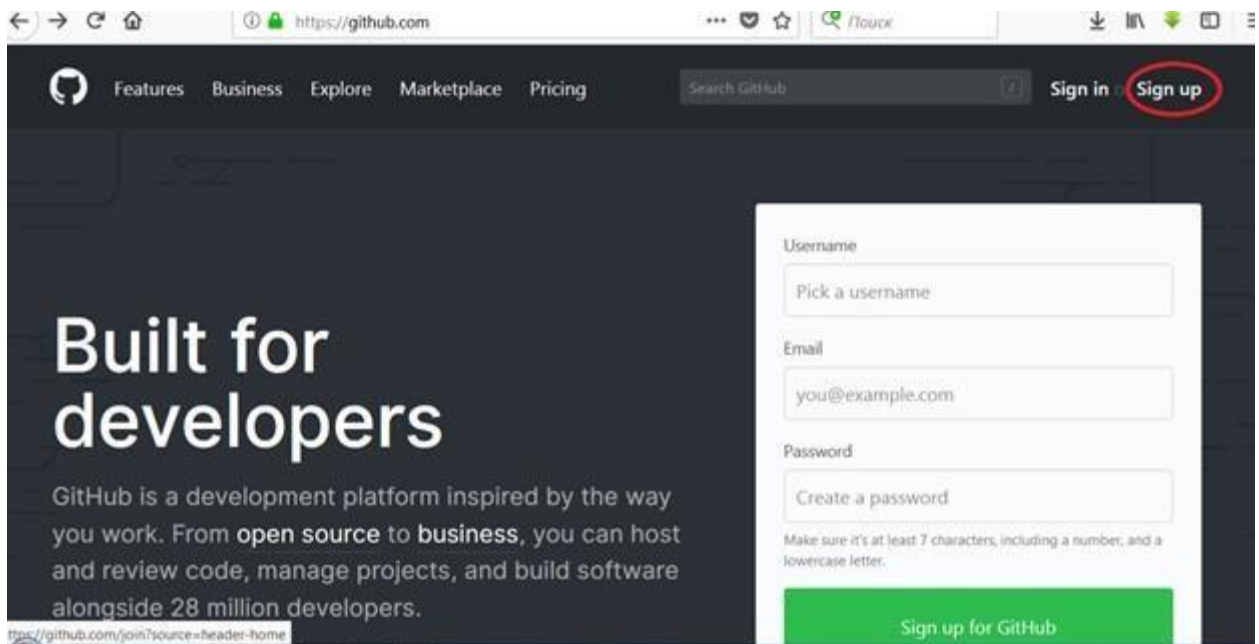
Основные шаги.

Шаг 1. Выбираем git-хостинг

Git-хостинг на разных условиях предлагают десятки компаний. Самые известные из них: Github, Sourceforge, Google Code, GitLab, Codebase. Выбери удобный интерфейс и регистрируйся на понравившемся хостинге. В этой статье мы рассмотрим работу с git-хостингом на примере Github’а.

Шаг 2. Регистрация

Процедура регистрации на Гитхабе простая и интуитивно понятная даже для тех, чей уровень английского далёк от Upper Intermediate.



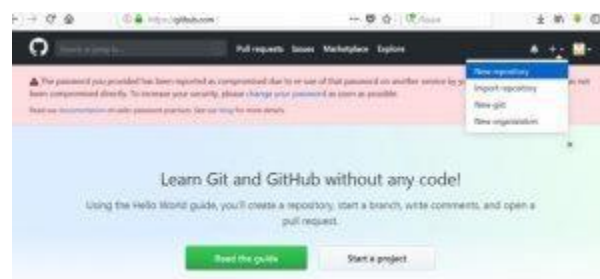
Логин, пароль, почта → подтверждение, и связь с мировым сообществом программистов налажена.

Шаг 3. Создание репозитория

Можно создать любое количество репозиториев, у каждого из которых будет issue tracking, wiki, условия для проведения code review и многое другое.

Политика разработчиков Github предполагает бесплатное использование хостинга для всех open-source проектов.

Чтобы создать новый репозиторий нажмём кнопку + в верхней части экрана и выберем New repository



Создание репозитория на Гитхабе

Многие разработчики рано или поздно сталкиваются с необходимостью создания приватного репозитория, код из которого доступен только их команде. Для этих случаев на Github'е есть определённый тарифный план.

Но пока острой необходимости в создании приватного репозитория у нас нет, создадим обычный.



Жмём волшебную кнопку Create внизу экрана, и репозиторий готов.

Шаг 4. Работа с репозиторием

Работа с репозиторием может вестись из командной строки, напрямую из среды разработки или из графического интерфейса (git — клиент приложения).

Работа с графическим интерфейсом позволяет лучше понимать процессы, происходящие в локальном и удалённом репозитории. Поэтому я рекомендую начать работу с git с использованием графического интерфейса.

Шаг 5. Выбираем Гит-клиент

Потом, когда суть процессов изменения и обновления (восстановления) информации в репозитории станет для Вас очевидна, можно работать и через командную строку. В этом принципе работы есть немало своих преимуществ. Например, все новые опции Гитхаба реализуются сначала для использования в командной строке, и только потом адаптируются под графические интерфейсы.

Но вернёмся к git-клиентам.

Самыми популярными гит- клиентами на данный момент являются:

SmartGit

Удобное приложение гармонично сочетает все необходимые функции и доступную интуитивно понятную систему управления. SmartGit – один из самых удобных графических интерфейсов для профессиональной разработки. Некоммерческая разработка и разработка open-source проектов не требуют платной лицензии.

GitHub Desktop

«Родной» графический интерфейс Гитхаба. GitHub Desktop работает под Windows и Mac и практически полностью копирует функционал основного сайта. Работает под той же учётной записью. Правда, не всегда оперативно

справляется с большими программами. Зато отлично подходит для начала работы с git.

GitKraken

Поддерживает GitHub, Bitbucket и Gitlab. Кракен очень любят программисты – фрилансеры, которым периодически приходится менять команды, а значит, и условия командной разработки. Возможность работы с разными git-хостингами через привычное приложение со знакомым интерфейсом в таких случаях играет важную роль.

SourceTree

SourceTree позволяет работать с Bitbucket и GitHub. В приложении довольно простой интерфейс, подходящий, как для опытных программистов, так и для новичков.

Шаг 6. Работа со SmartGit

Скачать SmartGit можно, например,

<https://www.syntevo.com/smartgit/download/>

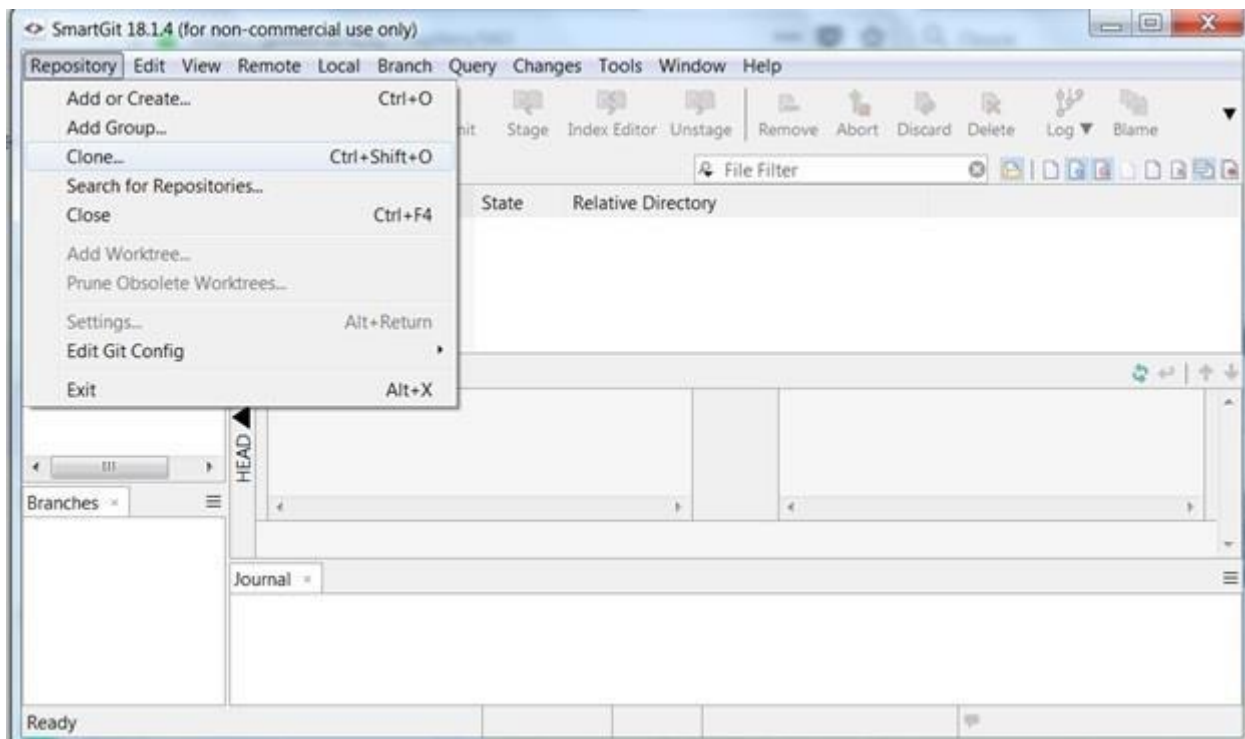
Устанавливаем и запускаем.

Основные операции для работы с git

Clone

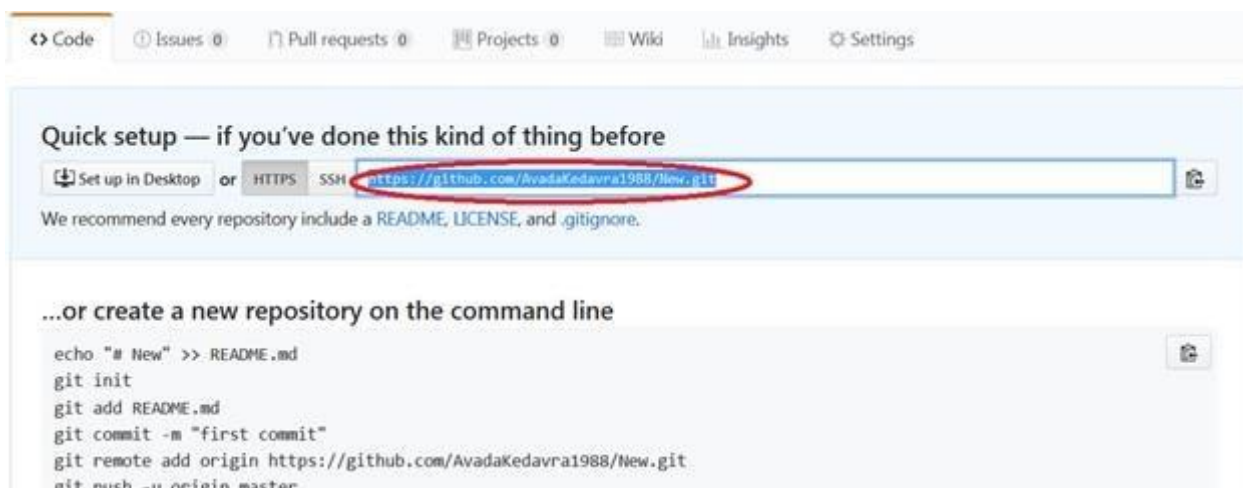
Первое, чему стоит научиться – это снимать копию проекта из удалённого репозитория в локальный.

Делается это довольно просто:



Clone

Затем копируем ссылку репозитория, созданного на Гитхабе (шаг 2)



Ссылка на репозиторий

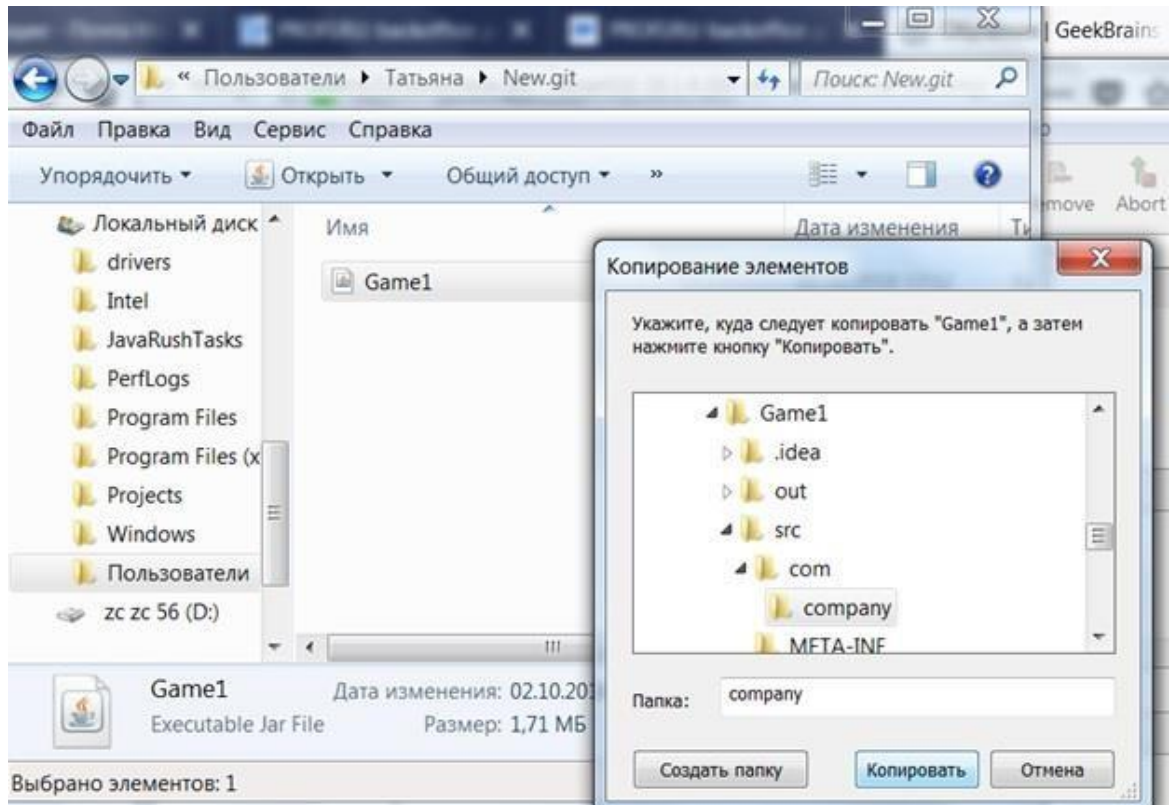
Вставляем адрес удалённого репозитория в нужную ячейку в открывшемся

окне, выбираем расположение нового локального репозитория у нас на компьютере, и получаем готовый локальный репозиторий.

К слову, аналогичным образом можно клонировать чужой открытый репозиторий и поближе познакомиться с чужим кодом.

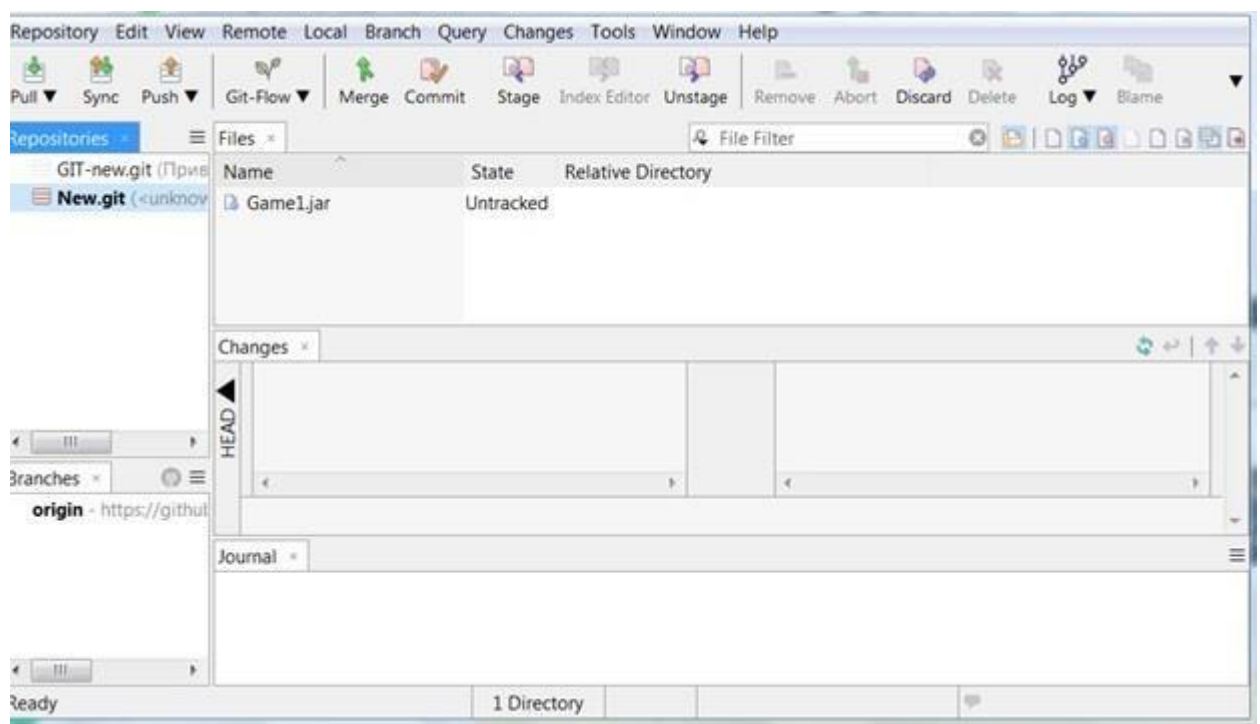
Commit

Репозиторий готов — пора приступать к работе. Написанный код мы помещаем в локальный репозиторий — папку .git (путь к которой мы указали в операции clone).



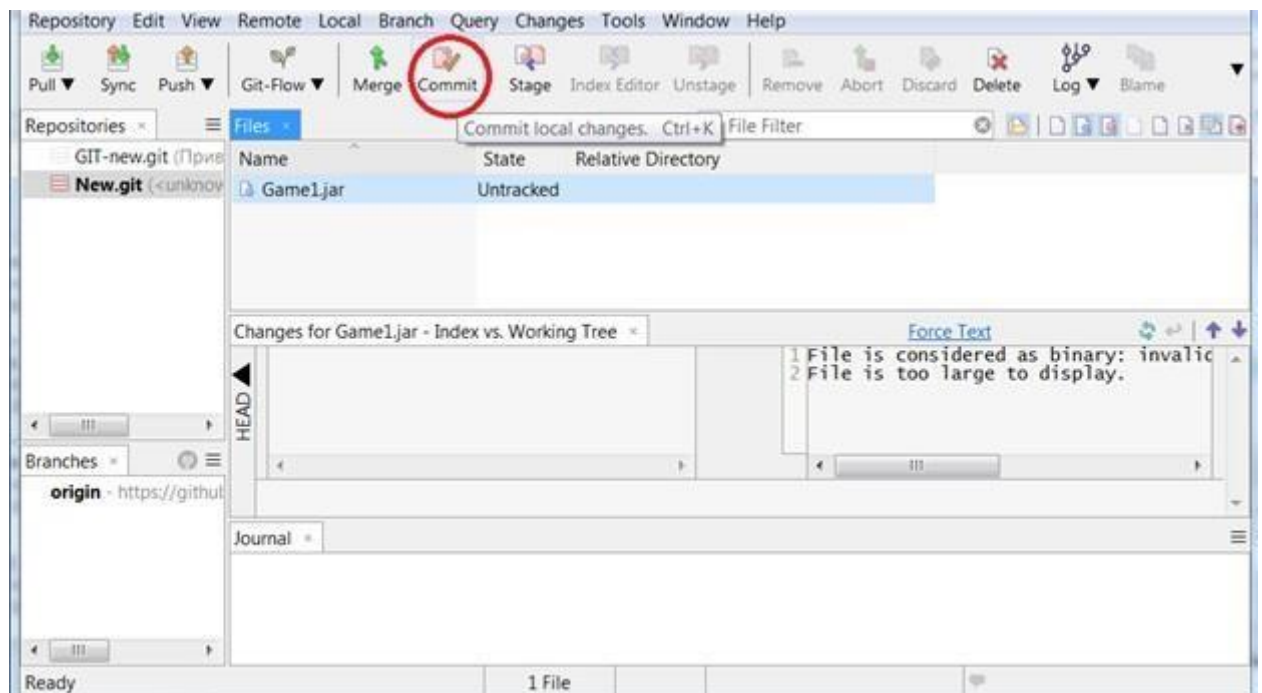
Добавление файла в локальный репозиторий.

Если всё прошло успешно, в окошке SmartGit'a появится скопированный файл.



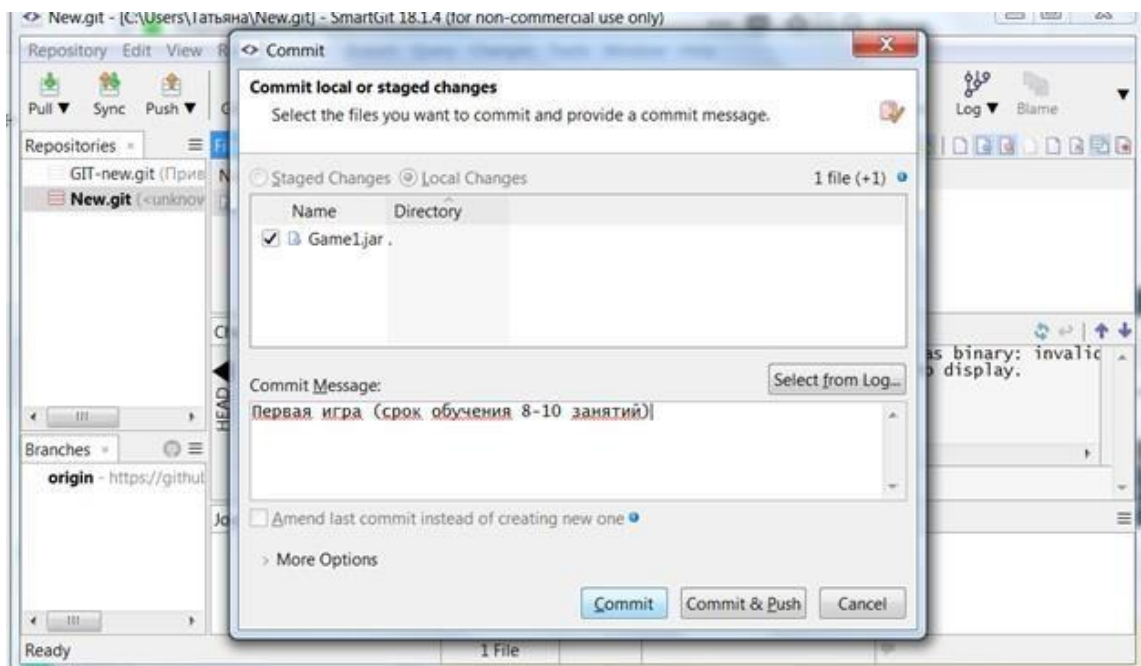
Новый файл в SmartGit.

Для того чтобы зафиксировать изменения в локальном репозитории, нажимаем кнопку Commit.



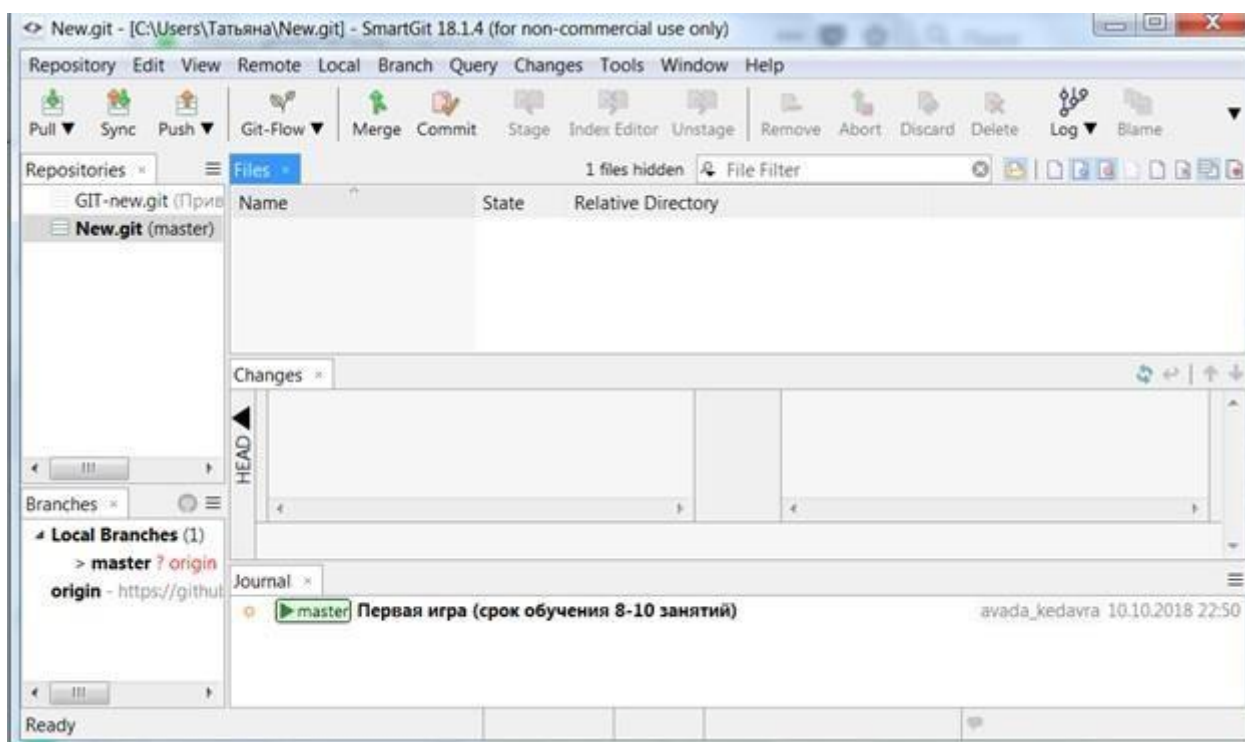
Commit

В открывшемся окне пишем пояснительный комментарий к сохраняемому файлу и снова нажимаем кнопку Commit



Пояснения к Commit'у

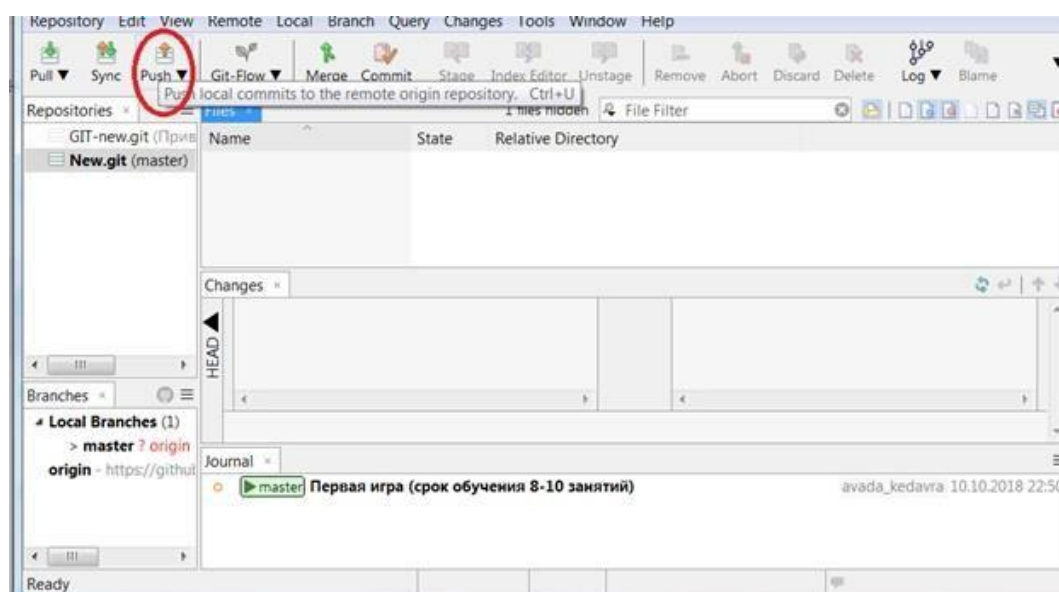
Файл сохранён, а изменения внесены в журнал.



Файл отправлен в локальный репозиторий

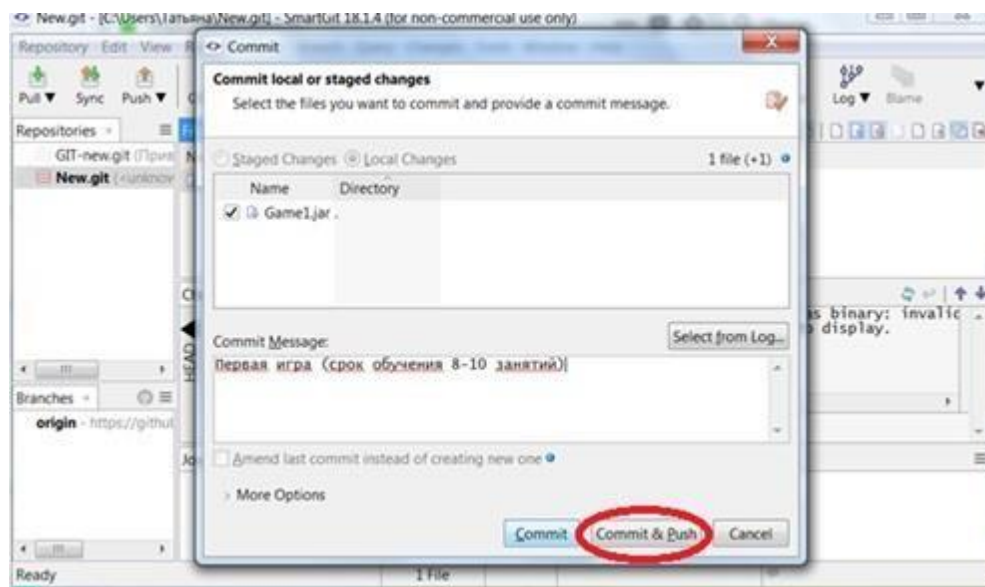
Push

Теперь заглянем на Github.com в наш удалённый репозиторий. Там до сих пор нет ни одного файла. Нужно срочно менять ситуацию. Чтобы перенести изменения, внесённые в локальный репозиторий, в удалённый репозиторий, необходимо нажать кнопку Push.



Push

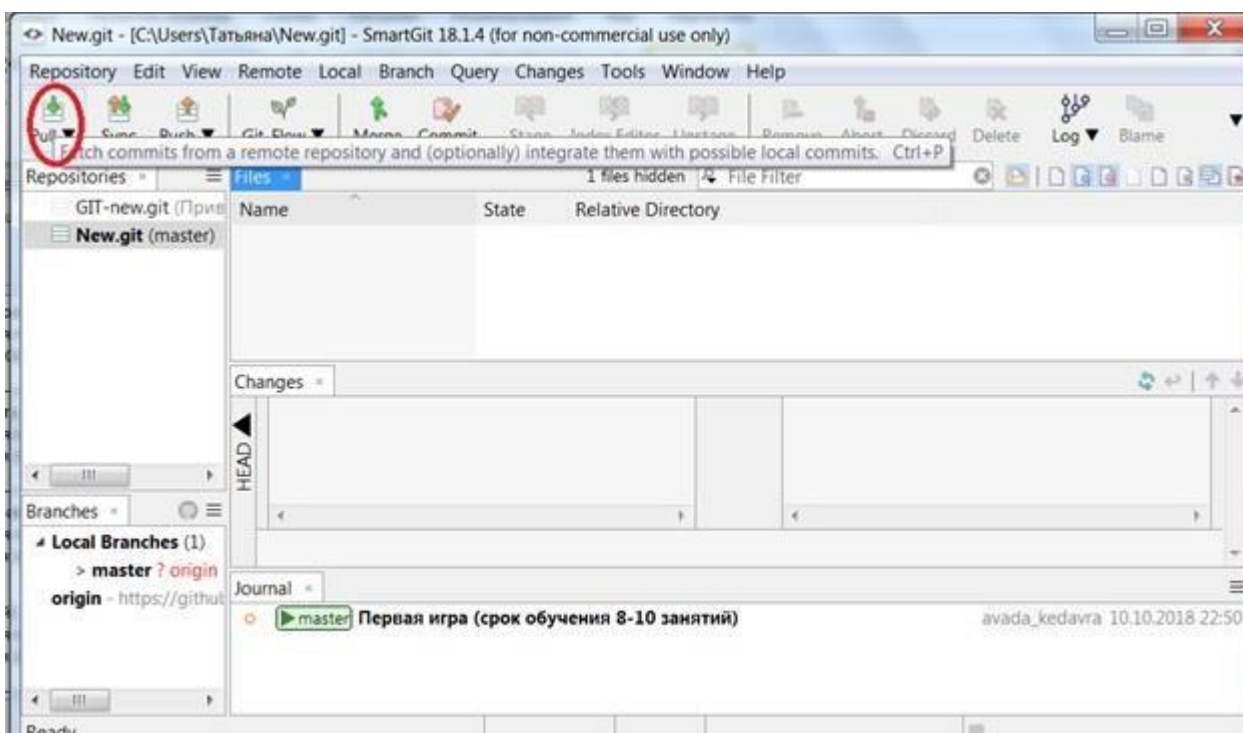
К слову, отправить изменения в удалённый репозиторий, нам предлагают ещё в точке Commit'a



Commit & Push

Pull

Возникает резонный вопрос: как получают изменения остальные участники разработки, если они клонировали проект в самом начале? Для этого существует команда Pull, передающая в локальный репозиторий все изменения, происходящие в удалённом.



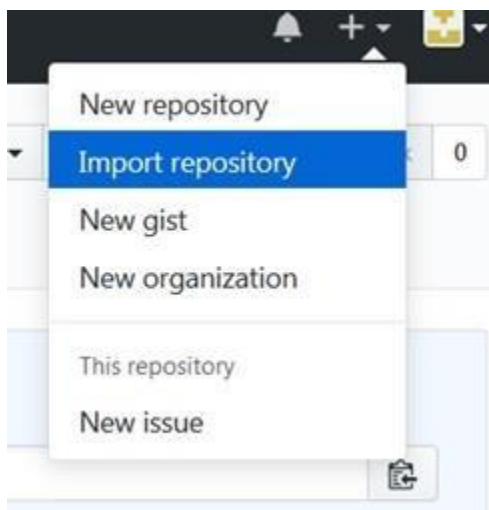
Pull

К слову, для командной разработки на Гитхабе есть ещё несколько важных

опций.

Перенос информации из сторонних репозиториев на Гитхаб

Когда нужно собрать разрозненные кусочки кода в один проект, используйте кнопку Import repository и работайте с файлами в удобном репозитории Гитхаба.



Импортировать репозиторий

Кнопка New gist на этой панели предназначена для мгновенного обмена информацией.

А кнопка New organization открывает массу возможностей для командной разработки.

Задание

1. Изучить теоретическую часть лабораторной работы
2. Пройти курс: <https://gb.ru/courses/1117>
3. Изучить теоретический материал курса <https://gb.ru/courses/1117> .
4. Написать отчет о прослушанном курсе с описанием выполненных заданий

Практическая работа № 6

Методологии управления ИТ-проектами

Цель : знакомство с методологиями управления ИТ-проектами.

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

В настоящее время управление проектами имеет свою методологию и основывается на определенных стандартах. В основе таких методов лежат методики сетевого планирования, которые появились США в конце 1950-х гг. При этом методики управления проектами широко распространились не только в странах с рыночной экономикой, но и в странах с т. н. «плановой» экономикой. Они начали использоваться в строительстве, что послужило основой их распространения в других отраслях и появлению методов проектного управления.

Примером здесь может служить объединение усилий фирмы «Дюпон» и фирмы «Ремингтон Рэнд» для составления плана графика комплексных работ по модернизации заводов «Дюпон» с помощью вычислительной машины Univac в 1956 г. Результатом стало создание рационального и простого метода описания проекта на ЭВМ – метода критического пути (CPM – Critical Path Method).

Практически параллельно и независимо был создан метод анализа и оценки программ PERT (Program Evaluation and Review Technique) в военно-морских силах США. Данный метод разрабатывался корпорацией «Локхид» и консалтинговой фирмой «Буз, Аллен энд Гамильтон» для ракетной системы «Поларис». Проект включал 3800 основных подрядчиков и состоял из 60 000 операций. Благодаря использованию данного метода проект стал успешным и завершился на два года раньше срока. После наглядного успеха метод PERT стал использоваться в вооруженных силах США для планирования проектов.

Задание: найдите примеры других крупных успешных проектов прошлого века, использующих для своего управления ЭВМ.

Учтите, что с одной стороны, применение информационных технологий для планирования и управления проектами давало в те времена значительное преимущество, а с другой стороны, первые компьютеры были дорогостоящими и оставались доступными только крупным компаниям. Все

изменилось с появлением персональных компьютеров, теперь ЭВМ стала рабочим инструментом для руководителей более широкого круга.

Система управления проектами постоянно развивалась и стала самостоятельной областью профессиональной деятельности. В итоге были созданы унифицированные методологии, инструментарии, механизмы, стандарты, доступные и для проектов в ИТ-сфере. Например, на сегодняшний день существует единая Международная ассоциация управления проектами – IPMA (International Project Management Association) с центром в г. Цюрих (Швейцария).

Из наиболее распространенных можно отметить процессную модель, используемую в документах методологических основ управления проектами – Project Management Body of Knowledge (PMBOK) Американского института управления проектами (PMI). Данный документ признается международным стандартом де-факто. Кроме того, стандарт ISO 10006:1997 придал ряду наиболее важных положений PMBOK статус стандарта де-юре.

В 2014 г. вышло пятое издание PMBOK, содержащее указания на 589 страницах. Все положения представлены на сайте <http://www.pmi.org/default.aspx>.

Данные Международной ассоциации управления проектами (IPMA) говорят о том, что использование современных методологий управления проектами экономит около 20–30% времени и около 15–20% средств на осуществление проектов.

Все методологии (еще их называют моделями, методиками) разработки программного обеспечения классифицируют по «весу», т. е. по количеству формализованных процессов и детальности их регламентации. Следовательно, чем больше процессов документировано, чем более детально описана методология, тем больше будет ее «вес».

1. Тяжеловесные методологии:

- ГОСТ 19 «Единая система программной документации» и ГОСТ 34 «Стандарты на разработку и сопровождение автоматизированных систем» ориентированы на последовательный подход к разработке программного обеспечения;

- Capability Maturity Model for Software (SW-CMM) определяет пять уровней «зрелости проекта»;

- Rational Unified Process (RUP) – итеративная модель разработки;
- Microsoft Solutions Framework (MSF) – база знаний компании Microsoft по разработке программ;
- Personal Software Process – модель определяет требования к компетенциям разработчика. Team Software Process – модель ориентирует на самоуправляемые команды от 3 до 20 разработчиков.
- и др.

2. Легковесные или agile-методики:

- eXtreme Programming или XP – экстремальное программирование, предлагающее 12 инженерных практик;
- Crystal Clear – семейство методологий, определяют необходимую степень формализации процесса разработки в зависимости от количества участников и критичности задач [13];
- Feature Driven Development (FDD) – функционально- ориентированная разработка;
- OpenUP – итеративно-инкрементальный метод разработки программ, позиционируется как легкий и гибкий вариант тяжеловесной методологии RUP;
- Scrum – управление разработкой информационных систем с высокой степенью неопределенности;
- Kanban – методология «бережливого производства»;
- и др.

Результаты исследования Agile Survey о популярности гибких методологий представлены на рисунок 1.

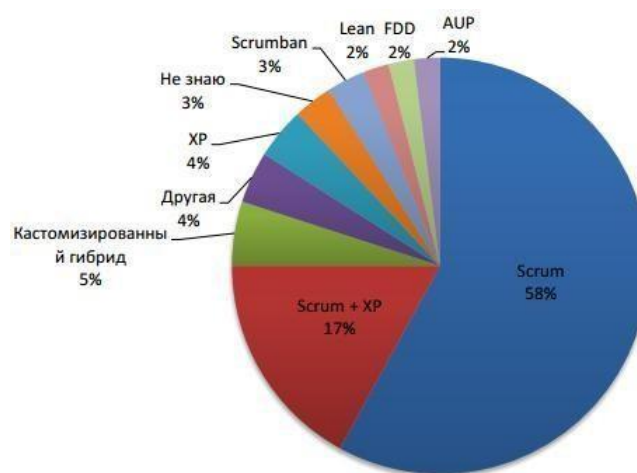


Рисунок 1 - Популярность гибких методологий

Легковесные методологии Agile появились сравнительно недавно. В феврале 2001 г. 17 специалистов (консультантов и практиков) провели семинар, на котором сформулировали основные принципы гибкой разработки ПО – Agile Manifesto – манифест гибкой разработки. Он переведен на многие языки мира и доступен на сайте <https://agilemanifesto.org/iso/ru/manifesto.html>.

Идея всех гибких (легковесных) методологий состоит в том, что применяемый в разработке ПО процесс должен быть адаптивным, ориентированным на людей и их взаимодействие. По сути это скорее набор практик, чем методология.

На том же семинаре было предложено новое название таких методологий – гибкая разработка Agile Software Development.

При выборе модели управления проектом можно ориентироваться на следующую таблицу (таблица 2).

Таблица 2 – Выбор методологии управления проектом

Вес модели	Достоинства	Недостатки
Тяжеловесные	– процессы рассчитаны на среднюю квалификацию исполнителей; – большая специализация исполнителей; – низкие требования к стабильности команды; – отсутствуют ограничения по срокам проектов.	– требуют существенной управленческой надстройки; – длительные стадии анализа и проектирования; – формализованные коммуникации.
Легковесные (гибкие)	– снижение непроизводительных расходов, связанных с управлением проектом, рисками, изменениями, конфигурациями; – упрощение стадий анализа и проектирования, основной упор на разработку функциональности, совмещение ролей; – неформальные коммуникации.	– эффективность сильно зависит от индивидуальных способностей, требуют более квалифицированной, универсальной и стабильной команды; – объем и сложность выполняемых проектов ограничены.

А. Коуберн, один из авторов «Манифеста», провел анализ ИТ-проектов за последние 20 лет [4], выполненных на основании разных моделей управления: от облегченных, гибких до тяжелых (СММ-5). Данный анализ показал отсутствие корреляции между успехом или провалом проектов и выбранными моделями разработки, применяемыми в проектах.

А. Коуберн сделал также вывод о том, что:

1. У каждого проекта должна быть своя модель процесса разработки.
2. У каждой модели – свое время.

Для разработчиков это означает, что не существует единственного правильного процесса разработки. В каждом новом проекте процесс должен определяться каждый раз заново по «закону четырех П» (рисунок 2).

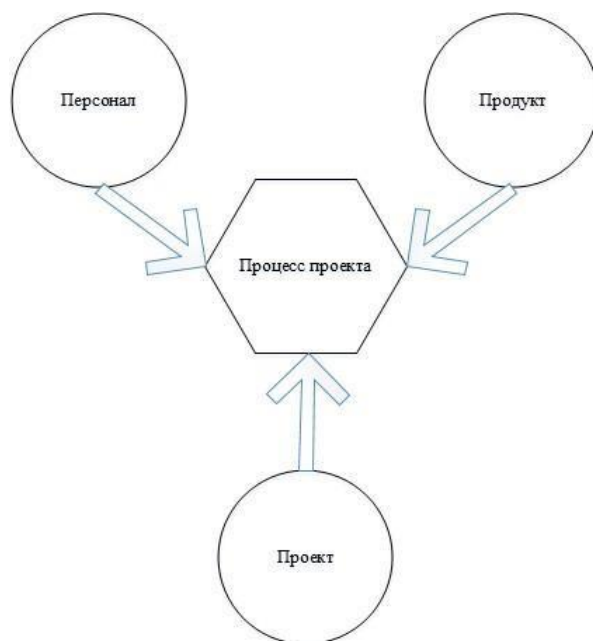


Рисунок 2 - Закон четырех П

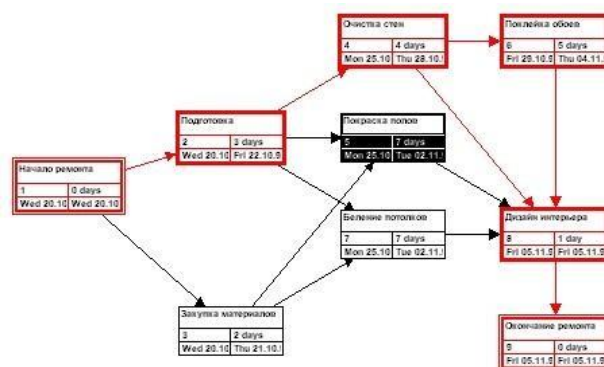


Рисунок 1.2 - ПЕРТ-диаграмма (PERT Chart)

Ход работы

Задание 1. С помощью поиска в сети Интернет найдите информацию о современных методологиях управления ИТ-проектами. Представьте основания для их классификации. Для каждого основания приведите примеры методологий.

Задание 2. Из полученного списка тяжеловесных методологий управления ИТ-проектами выберите один. Проведите исследование методологии. Результат представьте в таблице (таблица 3).

Таблица 3 – Особенности методики

Характеристика	Описание
Полное название методологии	
Авторы	
История возникновения	
Страна появления	
Основные принципы, подходы	
Имеются ли программные средства реализации методологии, какие?	
Используется ли в настоящее время	
Примеры успешных проектов, реализованных с помощью данной методологии	

Задание 3. Из полученного списка легковесных (agile) методологий управления ИТ-проектами выберите один. Проведите исследование методологии. Результат представьте в таблице (таблица 3).

Задание 4. Выберите любую из проанализированных методологий. Создайте о ней презентацию на 10-15 слайдов. Выступите в группе, будьте готовы ответить на вопросы.

Контрольные вопросы:

1. Что такое методология управления ИТ-проектом?
2. Какие виды методологий вы знаете?
3. В чем особенности тяжеловесных и легковесных методологий управления?
4. Приведите примеры методологий, используемых для разработки ИТ-проект

Практическая работа № 7

Изучение жизненного цикла программного продукта

Цель: Изучение жизненного цикла программного продукта. Ознакомление с различными моделями жизненного цикла продукта. изучение подходов к оценке стоимости программного обеспечения (ПО). Применение оценки стоимости этапа разработки ПО и для оценки стоимости сопровождения ПО.

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

Методология проектирования информационных систем описывает процесс создания и сопровождения систем в виде жизненного цикла (ЖЦ) ИС, представляя его как некоторую последовательность стадий и выполняемых на них процессов. Для каждого этапа определяются состав и последовательность выполняемых работ, получаемые результаты, методы и средства, необходимые для выполнения работ, роли и ответственность участников и т.д. Такое формальное описание ЖЦ ИС позволяет спланировать и организовать процесс коллективной разработки и обеспечить управление этим процессом.

Жизненный цикл ИС можно представить как ряд событий, происходящих с системой в процессе ее создания и использования.

Модель жизненного цикла отражает различные состояния системы, начиная с момента возникновения необходимости в данной ИС и заканчивая моментом ее полного выхода из употребления. Модель жизненного цикла - структура, содержащая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного продукта в течение

всей жизни системы, от определения требований до завершения ее использования.

В настоящее время известны и используются следующие модели жизненного цикла:

- **Каскадная модель** ([рис. 2.1](#)) предусматривает последовательное выполнение всех *этапов проекта* в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе.
- **Поэтапная модель с промежуточным контролем** ([рис. 2.2](#)). Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.
- **Спиральная модель** ([рис. 2.3](#)). На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка. Особое внимание уделяется начальным этапам разработки - анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (*макетирования*).



Рис. 2.1. Каскадная модель ЖЦ ИС



Рис. 2.2. Поэтапная модель с промежуточным контролем

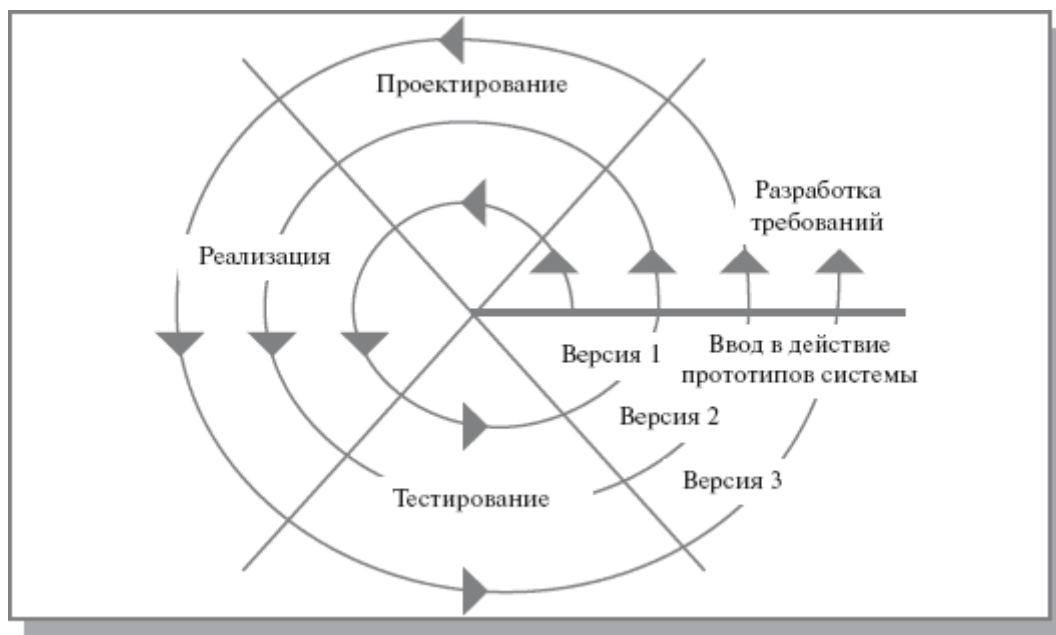


Рис. 2.3. Спиральная модель ЖЦ ИС

На практике наибольшее распространение получили две основные *модели жизненного цикла*:

- *каскадная модель* (характерна для периода 1970-1985 гг.);
- *спиральная модель* (характерна для периода после 1986 г.).

В ранних проектах достаточно простых ИС каждое *приложение* представляло собой единый, функционально и информационно независимый блок. Для разработки такого типа приложений эффективным оказался каскадный способ. Каждый этап завершался после полного выполнения и документального оформления всех предусмотренных *работ*.

Можно выделить следующие положительные стороны применения каскадного подхода:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логической последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадный подход хорошо зарекомендовал себя при построении относительно простых ИС, когда в самом начале разработки можно достаточно точно и полно сформулировать все требования к системе. Основным недостатком этого подхода является то, что реальный процесс создания системы никогда полностью не укладывается в такую жесткую схему, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ИС оказывается соответствующим *поэтапной модели с промежуточным контролем*.

Однако и эта схема не позволяет оперативно учитывать возникающие изменения и уточнения требований к системе. Согласование результатов разработки с пользователями производится только в точках, планируемых после завершения каждого этапа *работ*, а общие требования к ИС зафиксированы в виде технического задания на все время ее создания. Таким образом, пользователи зачастую получают систему, не удовлетворяющую их реальным потребностям.

Спиральная модель ЖЦ была предложена для преодоления перечисленных проблем. На этапах анализа и проектирования реализуемость технических решений и степень удовлетворения потребностей заказчика проверяется путем создания прототипов. Каждый виток спирали соответствует созданию работоспособного фрагмента или версии системы. Это позволяет уточнить требования, цели и характеристики проекта, определить качество разработки, спланировать работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который удовлетворяет действительным *требованиям заказчика* и доводится до реализации.

Итеративная разработка отражает объективно существующий спиральный цикл создания сложных систем. Она позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем и решить главную задачу - как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Основная проблема спирального *цикла* - *определение* момента перехода на следующий этап. Для ее решения вводятся временные ограничения на каждый из этапов *жизненного цикла*, и переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. Планирование производится на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Ход работы:

1. Изучить модели жизненного цикла продукта;
2. Типизация технологий разработки программ и различий в моделях жизненного цикла программного продукта;
3. Построение модели жизненного цикла своего программного продукта.
Определение необходимых ресурсов для его разработки.
4. Изучение подходов к оценке стоимости ПО;

Контрольные вопросы:

1. Что такое жизненный цикл продукта?
2. От чего зависят модель жизненного цикла?
3. Какие типы технологий разработки программ Вы знаете?
4. Какие виды ресурсов необходимы для разработки ПО?
5. Как определять ресурсы, необходимые для разработки продукта?
6. Какие подходы применяются к оценке стоимости ПО
7. Что такое технико-экономическое обоснование проекта?
8. Какая роль ТЭО и его отличие от бизнес-плана?

Практическая работа № 8

Знакомство с возможностями ARGOUML

Цель: изучить возможности программы ArgoUML и ее установку.

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

В последнее десятилетие прошлого тысячелетия, объектно-ориентированная (ОО) технология, наконец, завершила превращение из лабораторного инструмента 1960-х в основную парадигму программного обеспечения. Путь развития ОО технологии был длинным и трудным в первую очередь потому, что для ее применения был необходим значительный сдвиг в мышлении программистов, дизайнеров, разработчиков и других лиц, участвующих в жизненном цикле разработки программного обеспечения. Основной предпосылкой, лежащей в основе ОО технологии, является моделирование. ОО технология разработана и реализована по существу, как моделирование реального мира с помощью программного обеспечения артефактов. Эта посылка дает такие мощные результаты благодаря своей простоте. При анализе, разработке и внедрении ОО систем могут быть использованы идеи, изложенные на одном и том же языке. Это позволяет применять моделирование при разработке и испытаниях вместо того, чтобы в первую очередь на самом деле строить систему. Эта особенность в сочетании с возможностью проектировать системы на очень высоком уровне благодаря использованию опыта практиков позволила осуществить разработку и успешную реализацию более сложных систем, чем это было возможно ранее.

Принятие унифицированного языка моделирования (UML) в качестве стандартного языка для описания объектно-ориентированного подхода продолжило продвижение ОО технологии в мейнстрим.

Программа ArgoUML была задумана как инструмент описания окружающей среды для использования при анализе и проектировании объектно-ориентированных программных систем. В этом смысле программа ArgoUML похожа на многие коммерческие CASE-средства для моделирования программных систем. Но ArgoUML имеет ряд очень важных отличий многих из

этих инструментов:

- ArgoUML опирается на исследования в области когнитивной психологии и включает в себя ряд функций, которые способствуют повышению производительности за счет поддержки когнитивных (познавательных) потребностей разработчиков объектно-ориентированного программного обеспечения.
- ArgoUML поддерживает открытые стандарты - UML, XMI, SVG, OCL и другие. Поэтому ArgoUML продолжает развиваться и имеет преимущество перед коммерческими CASE-средствами.
- ArgoUML является 100%приложением Java. Это позволяет ArgoUML работать на всех платформах, для которых доступна платформа Java2.
- ArgoUML является программным продуктом с открытым исходным кодом. Открытый код гарантирует, что разработчики и исследователи программного обеспечения нового поколения теперь будут иметь проверенную основу, которая позволит им управлять развитием и эволюцией CASE- технологий.

ArgoUML является предметно-ориентированной средой разработки, что обеспечивает когнитивную поддержку объектно-ориентированного проектирования. ArgoUML обеспечивает те же функции автоматизации, что и коммерческие CASE-средства, но она ориентирована на поддержку когнитивных (познавательных) потребностей разработчиков. Эти познавательные потребности описаны в тех когнитивных теориях:

- 1) рефлексия-в-действии (reflection-in-action);
- 2) оппортунистический дизайн (opportunistic design);
- 3) понимание и решение проблем (comprehension and problem solving).

Первый выпуск ArgoUML стал доступен в 1998 году. С настоящее время программа ArgoUML стала популярной в образовательной и коммерческой сферах. Узнать больше о проекте Арго можно, используя ссылки

<http://argouml.tigris.org> и

<http://argouml.tigris.org/servlets/ProjectMailingListList>

Методика и порядок выполнения работы

Изучить теоретическое обоснование. Дополнительно можно изучить материалы о программе ArgoUML на сайте открытой энциклопедии Википедии. Для установки программы ArgoUML на домашнем компьютере можно бесплатно в Internet скачать ArgoUML для Windows.

Задание 1. Установить программу ArgoUML на том компьютере, который вы будете использовать для выполнения лабораторных работ. Если программа уже установлена, то убедитесь, что ее версия v0.34 или выше (рисунок 1.1).

При загрузке выводится одно из шести окон:

- Splash. (установлено по умолчанию). Выводит логотип ArgoUML и номер текущей версии (рисунок 8.1).
- Version. Выводит информацию о текущей версии, различных пакетах ArgoUML, некоторые сведения об операционной системе и среде разработки.
- Credits. Содержит информацию о создателях ArgoUML, включая детали о разработчиках отдельных модулей.
- Contact Info. Выводит контактную информацию об ArgoUML, адрес web-сайта.
- Report bugs. Выводит сообщения об ошибках.

После загрузки программы картинка (Splash), представленная на рисунке 1.1, исчезнет. Вместо нее появится интерфейс пользователя, описанный далее.



Рисунок 1.1 – Вид экрана во время загрузки ArgoUML

Содержание отчета и его форма

Отчет по практической работе должен состоять из описания основных этапов установки программы.

Контрольные вопросы:

1. Что такое «лицензионное соглашение»?
2. Какие виды лицензионных соглашений вы знаете?
3. Какие условия установки ArgoUML используются?

Практическая работа № 9

Визуальные средства представления проекта. Основы UML

Цель: освоить базовые приемы проектирования систем и процессов с использованием универсального языка моделирования (UML), получить навыки построения диаграмм классов UML.

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

Диаграмма классов.

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

Основными элементами являются классы и связи между ними. Классы характеризуются при помощи атрибутов и операций.



Рисунок 1 – Основные элементы диаграммы классов

Атрибуты описывают свойства объектов класса. Большинство объектов в классе получают свою индивидуальность из-за различий в их атрибутах и взаимосвязи с другими объектами. Однако, возможны объекты с идентичными значениями атрибутов и взаимосвязей. Т.е. индивидуальность объектов

определяется самим фактом их существования, а не различиями в их свойствах. Имя атрибута должно быть уникально в пределах класса. За именем атрибута может следовать его тип и значение по умолчанию.

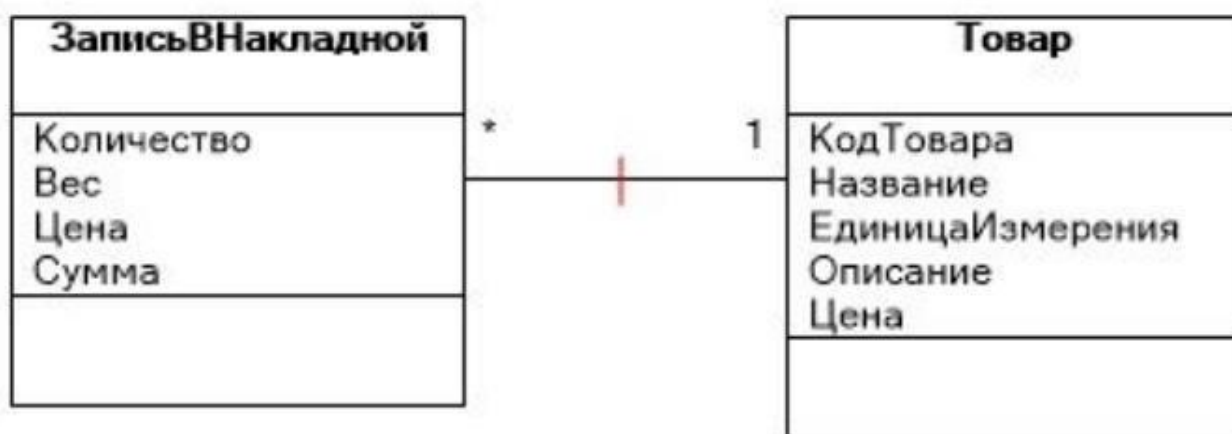
Операция есть функция или преобразование. Операция может иметь параметры и возвращать значения.

Виды связей:

1. ассоциация;
2. агрегация;
3. наследование.

Ассоциация (association) – представляет собой отношения между экземплярами классов. Каждый конец ассоциации обладает кратностью (синоним – мощностью, ориг. – multiplicity), которая показывает, сколько объектов, расположенных с соответствующего конца ассоциации, может участвовать в данном отношении. В примере на рисунке каждый Товар имеет сколь угодно Записей в накладной, но каждая Запись в накладной обязательно один Товар. В общем случае кратность может быть задана любым множеством.

Ассоциации может быть присвоено имя. В качестве имени обычно выбирается глагол или глагольное словосочетание, сообщающие смысл и назначение связи. Также, на концах ассоциации под кратностью может указываться имя роли, т.е. какую роль выполняют объекты, находящиеся с данного конца ассоциации.



Агрегация (aggregation) – это ассоциация типа «целое-часть».

Агрегация в UML представляется виде прямой с ромбом на конце.

Ромб на связи указывает, какой класс является агрегирующим (т.е. «состоящим из»), – класс с противоположного конца – агрегированным (т.е. те самые «части»).

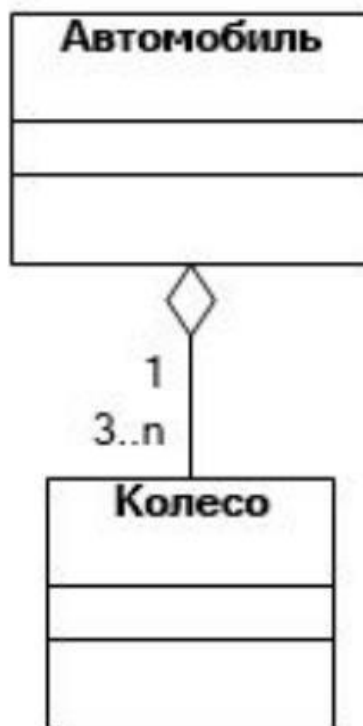


Рисунок 3 – Агрегация

Композиция (composition) – это такая агрегация, где объекты-части не могут существовать сами по себе и уничтожаются при уничтожении объекта агрегирующего класса. Композиция изображается так же, как ассоциация, только ромбик закрашен.



Рисунок 4 – Композиция

Важно понимать разницу между агрегацией и композицией: при

агрегации объекты-части могут существовать сами по себе, а при композиции – нет. Пример агрегации: автомобиль-колесо, пример композиции: дом-комната.

Наследование (inheritance) – это отношение типа «общее-частное».

Позволяет определить такое отношение между классами, когда один класс обладает поведением и структурой ряда других классов. При создании производного класса на основе базового (одного или нескольких) возникает иерархия наследования. Реализация принципов наследования является ключевой предпосылкой возможности повторного использования кода, поскольку это основной инструмент достижения полиморфизма.



Рисунок 5 – Наследование

Для построения диаграммы классов необходимо проанализировать предметную область и построить диаграмму классов. Должна получиться диаграмма, подобная следующему примеру.

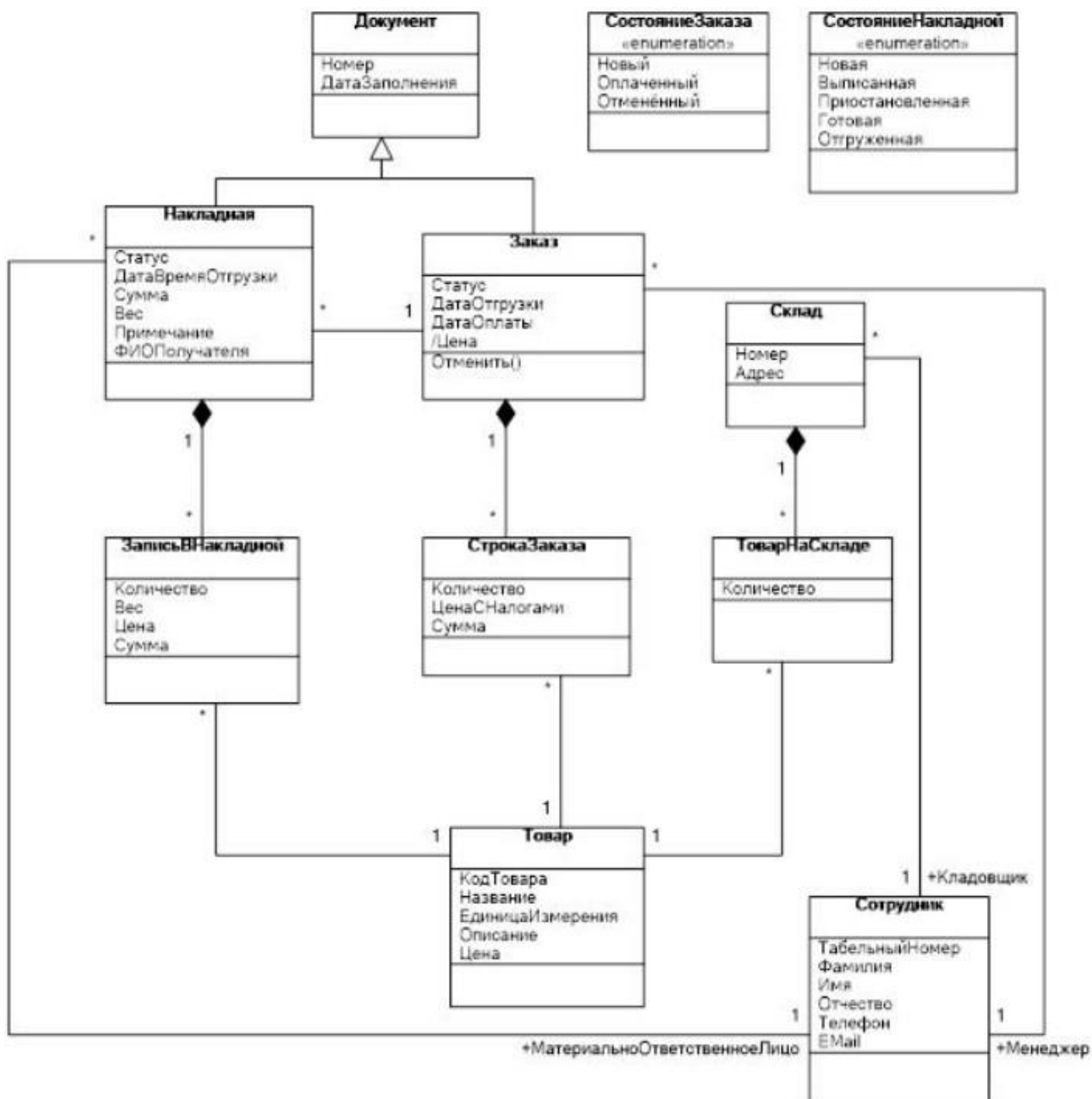


Рисунок 6 – Диаграмма классов

Основной сущностью в системе будет являться товар. Как известно из задания на проектирование, товар хранится на складе. Но понятия товара как некоего описания и товара, лежащего непосредственно на складе, отличаются друг от друга. Товар, лежащий на складе, кроме того, что связан со складом отношением композиции (агрегация не совсем подходит, поскольку в данной системе товар является товаром, пока он не покинет склад), ещё характеризуется количеством.

Аналогично следует рассуждать и при рассмотрении отношения Товара и Заказа, Товара и Накладной. В связи с тем, что Заказ и Накладная в сущности являются документами и имеют сходные атрибуты, они были объединены с помощью общего класса-предка Документ. Примечательно, что на диаграмме

Представлены два класса со стереотипом Enumeration (перечисление). Стереотип можно установить из контекстного меню для класса.

Дополнительная информация: <https://habr.com/ru/post/150041/>
<https://www.intuit.ru/studies/courses/1007/229/lecture/5956?page=1>

Методика и порядок выполнения работы

1. Практическая работа выполняется индивидуально.
2. Для выполнения практической работы рекомендуется использовать графический редактор Draw (<https://app.diagrams.net/>) или любой другой графический редактор, поддерживающий элементы UML.
3. Необходимо выбрать вариант задания для выполнения в рамках практической работы.

Предлагаемые варианты:

1. Диаграмма классов для CRM-системы для SMM-агентства;
2. Диаграмма классов для CRM-системы компании-производителя одежды;
3. Диаграмма классов для интернет-магазина крупного производителя игрушек;
4. Диаграмма классов для youtube-like видеосервиса на заказ: мобильное приложение + веб-приложение + админ панель;
5. Диаграмма классов для корпоративной социальной сети для нефтедобывающей компании;
6. Диаграмма классов для сервиса по продаже авиа и ж/д билетов.

Выбранную систему необходимо декомпозировать на составляющие элементы.

Для декомпозиции рекомендуется использовать технологию майндмеппинга и программные средства XMind или Mindmeister. Дополнительную информацию про использование технологии майндмеппинга можно найти здесь: <https://habr.com/ru/company/devexpress/blog/291028/>.

По результатам декомпозиции необходимо построить диаграмму классов для системы. Теоретическую информацию по построению диаграммы классов и пример диаграммы классов можно найти выше.

Оформить отчет с подробным описанием этапов выполнения работы

Контрольные вопросы

1. Что такое UML? Каковы его основные цели и области применения?

2. Перечислите основные типы диаграмм UML и их назначение.
3. Что такое диаграмма классов (Class Diagram)? Какие элементы она включает?
4. Что такое класс, объект и атрибут в UML? Как они изображаются на диаграммах?
5. Как обозначаются отношения между классами (ассоциация, агрегация, композиция, наследование)?
6. Как изображаются методы и атрибуты класса на диаграмме классов?
7. В чем разница между диаграммой классов и диаграммой объектов?
8. Как UML помогает в документировании программного проекта?
9. Какие проблемы могут возникнуть при использовании UML в больших проектах?
10. Какие инструменты для создания UML-диаграмм вы знаете? В чем их преимущества и недостатки?

Практическая работа № 10

UML. Диаграмма вариантов использования

Цель: применить диаграмму вариантов использования для описания требований к разрабатываемой системе. Ознакомиться с инструментами, позволяющими строить UML диаграммы.

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

Общая информация. Визуальным моделированием (visual modeling) называется способ представления идей и проблем реального мира с помощью моделей.

Модель — это абстракция, описывающая суть сложной проблемы или структуры без акцента на несущественных деталях, тем самым делая ее более понятной. Разработка программного обеспечения — не исключение. При построении сложной системы строятся ее абстрактные визуальные модели.

В настоящее время в области проектирования информационных систем с успехом применяется визуальное моделирование с помощью унифицированного языка моделирования UML.

С помощью UML можно детально описать систему, начиная разработку с концептуальной модели с ее бизнес функциями и процессами, а также описать особенности реализации системы, такие как классы программного обеспечения системы, схему базы данных.

Моделирование с помощью UML осуществляется поэтапным построением ряда диаграмм, каждая из которых отражает какую-то часть или сторону системы либо ее замысла.

Основные диаграммы UML:

1. вариантов использования (use case diagram);
2. классов (class diagram);
3. кооперации (collaboration diagram);

4. последовательности (sequence diagram);
5. состояний (statechart diagram);
6. деятельности (activity diagram);
7. компонентов (component diagram);
8. развертывания (deployment diagram).

Для сопровождения процесса построения, анализа и документирования модели, а также проверки модели и генерации программных кодов разработчики используют специально для этих целей созданные CASE-инструменты проектирования систем.

CASE (ComputerAided Software Engineering) — это набор инструментов и методов программной инженерии для проектирования программного обеспечения, цель которого — обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов.

Для выполнения лабораторных работ можно использовать любой графический редактор, поддерживающий нотацию UML, или специализированные CASE средства. Рекомендованным инструментом является свободное программное средство StarUML.

Диаграмма вариантов использования. Поведение системы (функции, которые она выполняет) описывают с помощью функциональной модели, которая отображает варианты использования (use cases, системные прецеденты), системное окружение (действующих лиц, актеров, actors) и связи между ними.

Диаграмма вариантов использования — это диаграмма, на которой изображаются отношения между актерами и вариантами использования.

С помощью этой диаграммы можно:

определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;

сформулировать общие требования к функциональному поведению проектируемой системы;

разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;

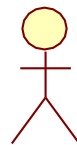
подготовить исходную документацию для взаимодействия разработчиков

системы с ее заказчиками и пользователями.

Диаграмма вариантов использования (прецедентов) представляет собой граф, в вершинах которого расположены актеры или прецеденты, связи между вершинами — это разного вида отношения.

Актером (действующее лицо, actor) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик.

На диаграммах вариантов использования актер изображается в виде человечка, под которым записывается его имя (рис. 1).



Преподаватель

Рисунок 1 — Действующее лицо

Вариант использования (прецедент, use case) — описание множества последовательности действий (включая варианты), выполняемых системой для того, чтобы актер мог получить определенный результат.

Каждый вариант использования должен быть независимым в том смысле, что если он всегда выполняется совместно с другим вариантом использования, то, скорее всего, это один прецедент, а не два, либо они связаны отношением включения или расширения, о чем речь пойдет позже.

Актер должен получить некоторый результат по завершению исполнения прецедента. Например, после исполнения прецедента в системе произойдут некоторые изменения: появятся новые данные, изменится поведение.

Каждый вариант использования описывает законченный процесс, переводящий систему из состояния готовности в новое состояние готовности. Вариант использования не может прерываться — он должен исполняться от начала до конца. По завершению одного варианта использования может начаться любой другой вариант использования (т. е. эта диаграмма не

показывает порядок действий, только набор возможных прецедентов).

Прецедент описывает, что делает система, но никак не уточняет, как она это делает.

На диаграмме прецедент изображается в виде эллипса. Имя прецедента может состоять из нескольких слов и знаков препинания (за исключением двоеточия), как правило, имя выбирают в виде словосочетания или глагольного выражения (рис. 2).

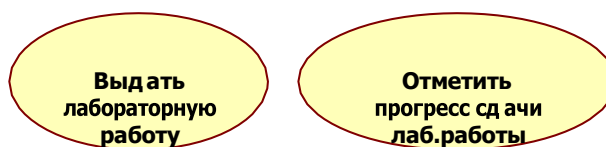


Рисунок 2 — Варианты использования

Одним из наиболее важных (и дорогостоящих) этапов проектирования информационных систем является этап определения требований к системе. Если требования заказчика информационной системы разработчиками будут определены не корректно, то в итоге заказчик может получить совсем не ту систему, которую он ожидал.

Моделирование прецедентов и актеров помогает нам лучше понять требования, предъявляемые к системе, и согласовать их с заказчиком с помощью демонстрации и обсуждения диаграммы прецедентов. Прецеденты и актеры — это отражение требований к системе, они показывают, кто и для чего будет использовать будущую систему.

Рассмотрим пример. Требуется разработать комплекс

«Автоматизированное рабочее место преподавателя». В зада и комплекса входит обеспечение следующих возможностей:

- составление и редактирования графика лабораторных работ;
- отмечание успешных и проваленных попыток сдачи лабораторных работ студентами;
- указывать темы РГР, курсовых работ и проектов;
- следить за текущей успеваемостью в разрезе группы, курса (для заведующего кафедрой) и факультета (для декана);

- отмечать пропуски на лабораторных занятиях и лекциях;
- запрашивать списки групп из автоматизированной системы «Деканат».

Проанализировав постановку задачи, приходим к выводу, что будет полезно выделить следующие действующие лица:

- преподаватель — преподаватель, ведущий практические занятия и лабораторные;
- лектор — преподаватель, ведущий лекции;
- руководитель — заведующие кафедрами, деканы;
- ассистент — ответственный за ввод данных в систему;
- АС «Деканат» — сервис автоматизированной системы «Деканат».

Обратите внимание, что актер — это роль, а не конкретный представитель этой роли. Например, в разное время заведующий кафедрой может выступать в роли лектора, преподавателя, ведущего практические занятия или руководителя.

Теперь выделим варианты использования на основе перечисленных функциональных требований:

- работа с графиком лабораторных работ;
- отмечание успеваемости;
- указание тем РГР, курсовых работ и проектов;
- отмечание посещения лекций;
- отмечание посещения лаб.работ;
- загрузка списков групп;
- создание лабораторных работ;
- создание РГР, курсовых работ и проектов;
- построение отчётов успеваемости и посещаемости.

Отношения

Связи и взаимоотношения, существующие между элементами модели, в UML описываются с помощью отношений, изображаемых на диаграммах.

Отношение (relationship) — это семантическая связь между отдельными элементами модели.

Между актерами и прецедентами диаграммы вариантов использования могут существовать отношения, показывающие, что экземпляры действующих лиц и вариантов использования взаимодействуют друг с другом.

Действующие лица могут находиться в отношениях с несколькими прецедентами и между собой, равно как и прецеденты могут быть связаны между собой отношениями особого рода.

Всего на диаграммах вариантов использования допустимы следующие типы отношений:

- обобщения (generalization relationship) — отношение между двумя актерами, либо двумя вариантами использования;
- ассоциации (association relationship) — отношение между актером и вариантом использования;
- включения (include relationship) — отношения между вариантами использования;
- расширения (extend relationship) — отношения между вариантами использования.

Ассоциация — это структурное отношение, показывающее, что объект неким образом связан с другим объектом. Отношение этого типа используется не только на диаграммах прецедентов, но и на других диаграммах. Ассоциативное отношение может быть направленным. В этом случае направление связи показывает, кто является инициатором (актер или разрабатываемая система). Например, если отношение направлено от актера к прецеденту, это означает, что актер инициирует выполнение прецедента.

Обобщение (Generalization) — это отношение между общей сущностью и ее конкретным воплощением. На диаграммах обобщение обозначается стрелкой с незакрашенным треугольником на конце, направленной от частного элемента к общему. По смыслу это отношение очень похоже на отношение

«является родителем» в ООП.

Отношения включения и расширения связаны с дополнительной информацией, сопутствующей диаграмме вариантов использования.

Для более подробного (больше, чем имя) описания элементов модели

можно использовать следующие подходы:

- краткое текстовое описание (в поле Documentation);
- связанные диаграммы (контекстное меню Add diagram) — диаграммы, более подробно описывающие конкретный элемент модели (например, алгоритм действий в конкретном варианте использования);
- вложения (Attachments) — прикрепленные документы (например, *.docx содержащие подробные описания).

Для описания элементов диаграммы вариантов использования часто используют:

- связанные диаграммы последовательностей и коопераций;
- вложенные документы — сценарии использования.

Сценарии применяются для текстового описания потоков событий. Поток событий — это определенная последовательность действий, которая описывает действия актеров и поведение моделируемой системы в форме текста.

Обычно потоки событий записываются в виде пронумерованной последовательности действий на естественном языке (хотя иногда применяют и псевдокод).

Каждый сценарий состоит из описания основного потока событий, его вариантов и исключительного потока.

Основной поток событий описывает ожидаемый вариант происходящих событий (например, все данные оказались правильными, файлы были найдены, места на сервере хватило).

Альтернативные потоки описывают различные варианты отклонения от обычного потока (например, выборка списка товара по дате, а не виду) или необязательные шаги (выбор цвета товара дополнительно к цене).

Исключительный поток показывает поведение системы в случае возникновения ошибок.

Если описание потока событий занимает много места, оно теряет выразительность. В таких случаях применяют следующие приёмы.

Если часть потока событий повторяется в нескольких вариантах использования, эту часть выносят в отдельный вариант использования. Прецеденты, из которых удалили эту часть, соединяют с новым вариантом

использования отношением

«включает». В описании включающих прецедентов указывается, что часть потока находится в другом прецеденте.

Необязательные этапы потока событий можно вынести в отдельный прецедент. Тогда этот новый прецедент будет

«расширять» оригинальный. В описании оригинального потока указывается точка расширения — место в котором могут

«подцепляться» прецеденты для расширения, а в расширяющих прецедентах указывается к какой именно точке расширения они относятся.

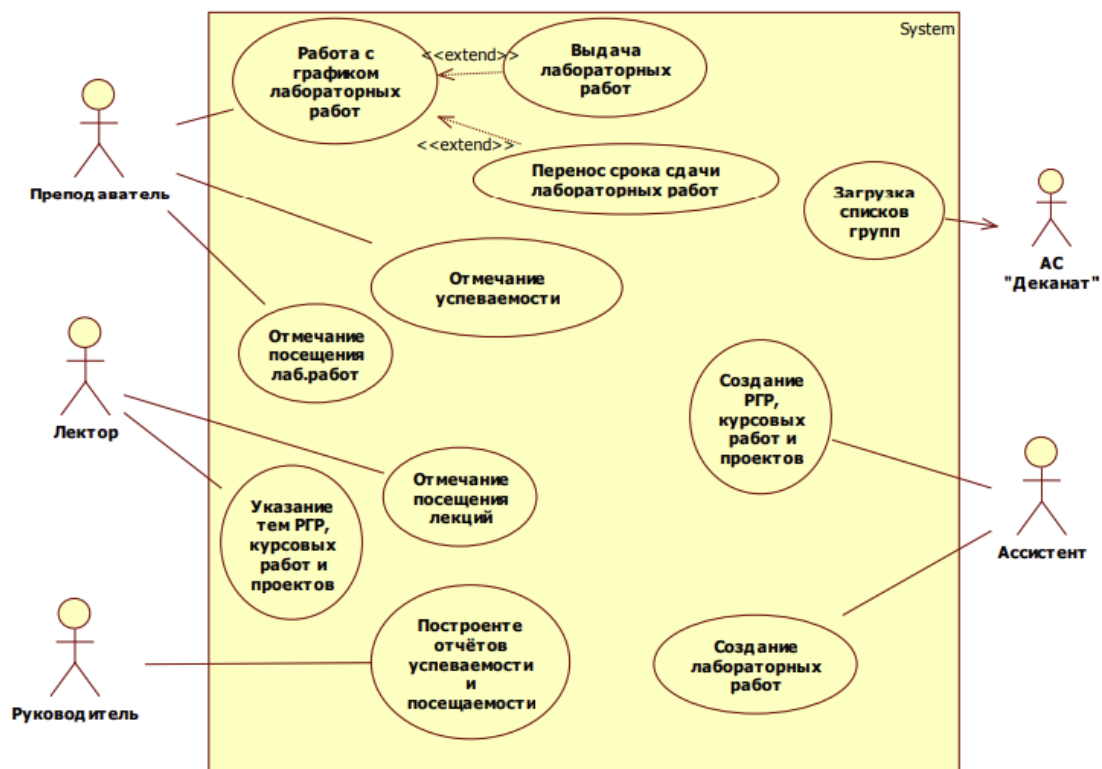


Рисунок 3 — Диаграмма вариантов использования
«АРМ Преподавателя»

Фрагмент сценария к варианту использования «Работа с графиком лабораторных работ».

1. Пользователь выбирает дисциплину.
2. Система показывает график выдачи и срока сдачи всех лабораторных работ по выбранному предмету.
3. Точка расширения «Работа с лабораторными работами».

4. Пользователь нажимает «вернуться назад».
5. Система показывает экран работы с дисциплиной.
6. Варианты использования «Перенос срока сдачи лабораторных работ» и «Выдача лабораторных работ» расширяют этот вариант использования в точке «Работа с лабораторными работами», что должно быть указано в их сценарии.
7. Точка расширения «Работа с лабораторными работами».
8. Сроки выдачи и сдачи лабораторных работ показаны в виде прямоугольников на оси времени.
9. Пользователь перетягивает один из углов прямоугольника, изображающего срок сдачи работы.
10. Срок сдачи лабораторной работы изменяется на графике.
11. Система показывает экран работы с дисциплиной.
12. Пользователь выбирает пункт «График лабораторных работ».

Индивидуальные задания

Выполнить задание в соответствии с темой исследования. В отчёт должны входить:

- одна или несколько диаграмм вариантов использования по индивидуальной теме;
- сценарии двух или трёх вариантов использования в виде текстового описания или псевдокода (основной, альтернативные и исключительные потоки событий);
- 2–3 актеров;
- 8–10 вариантов использования;
- желательно использование отношений включения, расширения, обобщения.

Контрольные вопросы

1. Что называют Визуальным моделированием?
2. Что такое модель?
3. Перечислите основные диаграммы UML.
4. Какие элементы входят в диаграмму вариантов использования (акторы, прецеденты)?

5. Как обозначаются отношения между акторами и прецедентами?
6. Для чего применяются диаграммы использования?

Практическая работа № 11

Построение диаграмм деятельности

Цель: Ознакомиться с методологией моделирования деятельности на основе языка UML

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

Диаграммы деятельности (Activity diagram), называемые также диаграммами активности или диаграммами видов деятельности, были введены в язык UML сравнительно недавно. Диаграмма деятельности - это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Результат может привести к изменению состояния системы или возвращению некоторого значения.

Диаграмма деятельности отличается от традиционной блок-схемы более высоким уровнем абстракции;

возможностью представления с помощью диаграмм деятельности управления параллельными потоками наряду с последовательным управлением.

Основными направлениями использования диаграмм деятельности являются визуализация особенностей реализации операций классов;

отображение внутрисистемной точки зрения на прецедент.

В последнем случае диаграммы деятельности применяют для описания шагов, которые должна предпринять система после того, как инициирован прецедент

особенности алгоритмической и логической реализации выполняемых системой операций и прецедентов;

выделить последовательные и параллельные потоки управления;

подготовить детальную документацию для взаимодействия разработчиков системы с ее заказчиками и проектировщиками.

Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия или

состояния деятельности, а дугами - переходы от одного состояния действия/деятельности к другому. Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояния (на практике иногда можно видеть несколько конечных состояний на одной диаграмме, но это одно и то же состояние, изображенное несколько раз для лучшей читабельности диаграммы). Саму диаграмму деятельности принято располагать таким образом, чтобы действия следовали сверху вниз. В этом случае начальное состояние будет изображаться в верхней части диаграммы, а конечное - в ее нижней части.

Рассмотрим основные элементы диаграммы деятельности.

Состояние деятельности (Activity, Process) - это **продолжающийся во времени неатомарный шаг вычислений в автомате**. Состояния деятельности могут быть подвергнуты дальнейшей декомпозиции, вследствие чего выполняемую деятельность можно представить с помощью других диаграмм деятельности. Состояния деятельности не являются атомарными, то есть могут быть прерваны. Предполагается, что для их завершения требуется заметное время.

Состояния действия (action state) - **состояние, которое представляет вычисление атомарного действия**, как правило - вызов операции. Состояния действия не могут быть подвергнуты декомпозиции. Они атомарны, то есть внутри них могут происходить различные события, но выполняемая в состоянии действия работа не может быть прервана. И наконец, обычно предполагается, что длительность одного состояния действия занимает неощутимо малое время. Действие может заключаться в вызове другой операции, послыке сигнала, создании или уничтожении объекта либо в простом вычислении - скажем, значения выражения.

Состояния деятельности и состояния действия имеют одинаковое стандартное графическое обозначение - прямоугольник с закругленными краями. Внутри такого символа записывают произвольное выражение (action-expression), которое должно быть уникальным в пределах одной диаграммы деятельности.

Начальное и конечное состояния на диаграммах деятельности изображаются как закрашенный кружок и закрашенный кружок внутри окружности, соответственно.

Ниже приведен пример диаграммы деятельности:



Рисунок 1 – Пример диаграммы деятельности

Переход (Transitions) - отношение между двумя состояниями, показывающее, что объект, находящийся в первом состоянии, должен выполнить некоторые действия и перейти во второе состояние. Когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока и используются переходы, показывающие путь из одного состояния действия или деятельности в другое. В UML переход представляется простой линией со стрелкой.

Ветвления. Простые последовательные переходы встречаются наиболее часто, но их одних недостаточно для моделирования любого потока управления. Как и в блок-схему, в диаграмму деятельности может быть включено *ветвление или множественный переход со сторожевыми условиями*. Ветвление описывает различные пути выполнения в зависимости от значения некоторого булевского выражения. Графически точка ветвления представляется ромбом. В точку ветвления может входить ровно один переход, а выходить

- два или более. Для каждого исходящего перехода задается булевское выражение, которое вычисляется только один раз при входе в точку ветвления. Ни для каких двух исходящих переходов сторожевые условия не

должны одновременно принимать значение "истина", иначе поток управления окажется неоднозначным. Но эти условия должны покрывать все возможные варианты, иначе поток остановится.

Разделения и слияния. Простые и ветвящиеся последовательные переходы в диаграммах деятельности используются чаще всего. Однако часто возникает потребность изображения параллельных потоков, и это особенно характерно для моделирования бизнес-процессов. В UML для обозначения разделения и слияния таких параллельных потоков выполнения используется синхронизационная черта, которая рисуется в виде жирной вертикальной или горизонтальной линии. При этом *разделение* (*concurrent fork*) имеет один входящий переход и несколько выходящих, *слияние* (*concurrent join*), наоборот, имеет несколько входящих переходов и один выходящий.

Пример разделения и слияния потоков приведен ниже:

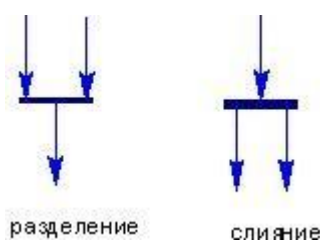


Рисунок 2 – Пример разделения и слияния потоков

Дорожки. При моделировании течения бизнес-процессов иногда бывает полезно разбить состояния деятельности на диаграммах деятельности на группы, каждая из которых представляет отдел компании, отвечающий за ту или иную работу. В UML такие группы называются дорожками (Swimlanes), поскольку визуально каждая группа отделяется от соседних вертикальной чертой, как плавательные дорожки в бассейне (см. рис. 3). Дорожки - это разновидность пакетов, описывающие связанную совокупность работ.

Каждой присутствующей на диаграмме дорожке присваивается уникальное имя. Никакой глубокой семантики дорожка не несет, разве что может отражать некоторую сущность реального мира. Каждая дорожка

представляет сферу ответственности за часть всей работы, изображенной на диаграмме. На диаграмме деятельности, разбитой на дорожки, каждая деятельность принадлежит ровно одной дорожке, но переходы могут пересекать границы дорожек.

Имеется некоторая связь между дорожками и параллельными потоками выполнения. Концептуально деятельность внутри каждой дорожки обычно - но не всегда - рассматривается отдельно от деятельности в соседних дорожках. Это разумно, поскольку в реальном мире подразделения организации, представленные дорожками, как правило, независимы и функционируют параллельно.

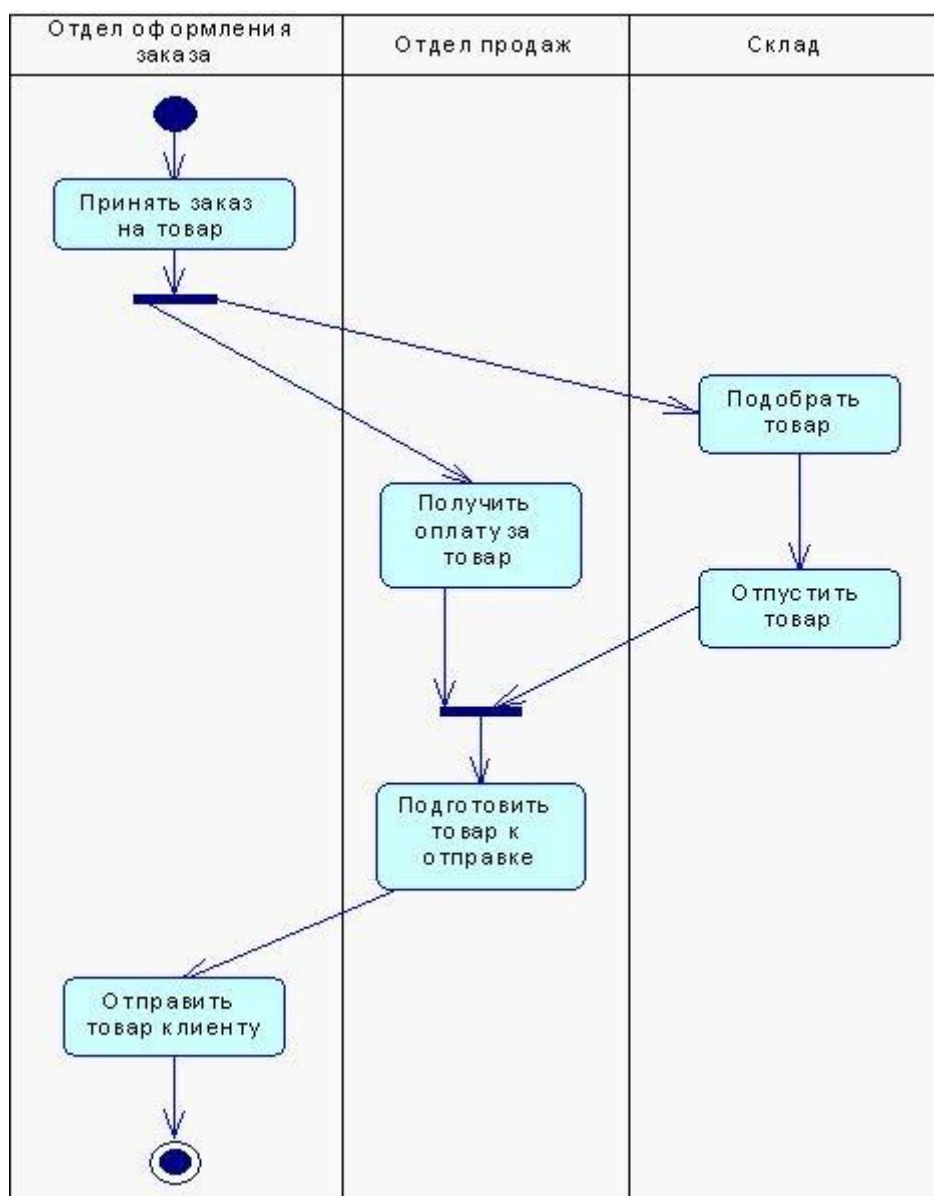


Рисунок 3 – Пример диаграммы деятельности для магазина видеопродукции

На следующем рисунке приведена диаграмма деятельности прецедента "Прокат видео" пример "Магазин видеопродукции"

Последовательность выполнения лабораторной работы:

1. Ознакомиться с методологией моделирования деятельности на основе языка UML.
2. Построить диаграммы деятельности для каждого прецедента присутствующего на диаграмме прецедентов, разработанной при выполнении лабораторной работы 3.
3. Оформить отчет.

Практическая работа № 12

Интерфейсы. Диаграммы компонентов

Цель: применить диаграмму компонентов для описания требований к разрабатываемой системе. Ознакомиться с инструментами, позволяющими строить UML диаграммы.

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Теоретическое обоснование

Интерфейс(interface) служит для спецификации параметров модели, которые видимы извне, без указания их внутренней структуры. В языке UML интерфейс является классификатором и характеризует только ограниченную часть поведения моделируемой сущности. Применительно к диаграммам вариантов использования, интерфейсы определяют совокупность операций, которые обеспечивают необходимый набор сервисов для актеров.

На диаграмме вариантов использования интерфейс изображается в виде *маленького круга*, рядом с которым записывается его имя. В качестве имени может быть существительное или строка текста. Если имя записывается на английском языке, то оно должно начинаться с заглавной буквы **I**.

Графический символ отдельного интерфейса соединяется на диаграмме сплошной линией или пунктирной линией со стрелкой с тем вариантом использования, который его поддерживает. Сплошная линия указывает, что связанный с интерфейсом вариант использования должен реализовывать все необходимые для него сервисы. Пунктирная линия со стрелкой означает, что вариант использования предназначен для спецификации только того сервиса, который необходим для реализации данного интерфейса.

Таким образом, интерфейс отделяет спецификацию операций системы от их реализации и определяет общие границы проектируемой системы.

Пример выполнения работы

Для представления физических сущностей в языке UML применяется специальный термин – компонент (component). Компонент реализует некоторый набор интерфейсов и служит для общего обозначения элементов физического представления модели. Для графического представления компонента может использоваться специальный символ – прямоугольник со вставленными слева двумя более мелкими прямоугольниками (рис. 1). Внутри объемлющего прямоугольника записывается имя компонента и, возможно, некоторая дополнительная информация.

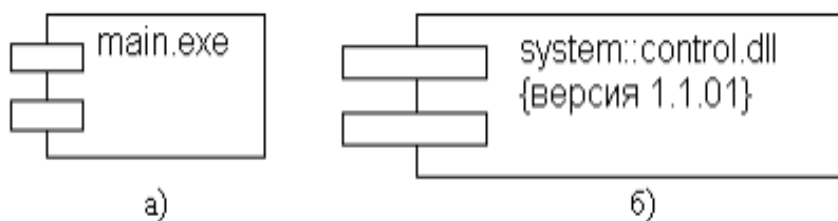


Рисунок 1 - Графическое изображение компонента в языке UML

В первом случае (рис. 1, а) с компонентом уровня экземпляра связывается только его имя, а во втором (рис. 1, б) – дополнительно имя пакета и помеченное значение. В языке UML выделяют три вида компонентов:

- компоненты развертывания, которые обеспечивают непосредственное выполнение системой своих функций: динамически подключаемые библиотеки с расширением dll, Web-страницы на языке разметки гипертекста с расширением html и файлы справки с расширением hlp.

- компоненты-рабочие продукты: файлы с исходными текстами программ, например, с расширениями h или cpp для языка C++.
- компоненты исполнения, представляющие исполнимые модули – файлы с расширением exe.

Следующим элементом диаграммы компонентов являются интерфейсы. Этот элемент уже рассматривался ранее, поэтому отметим только его особенности, которые характерны для представления на

диаграммах компонентов. В общем случае интерфейс графически изображается окружностью, которая соединяется с компонентом отрезком линии без стрелок (рис. 2, а). Семантически линия означает реализацию интерфейса, а наличие интерфейсов у компонента означает, что данный компонент реализует соответствующий набор интерфейсов.

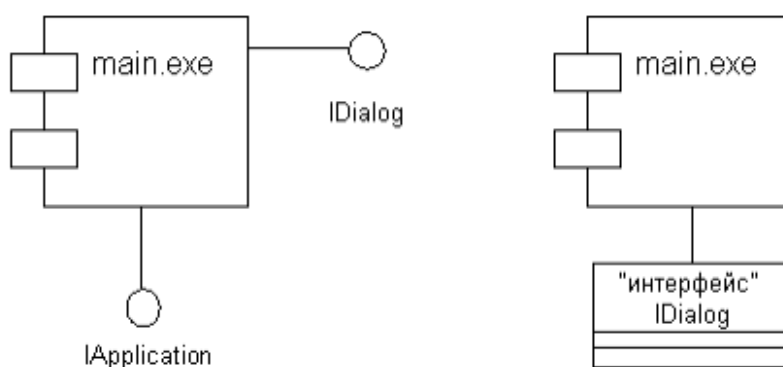


Рисунок 2 - Графическое изображение интерфейсов на диаграмме компонентов

Другим способом представления интерфейса на диаграмме компонентов является его изображение в виде прямоугольника класса со стереотипом "интерфейс" и возможными секциями атрибутов и операций (рис. 2, б). Как правило, этот вариант обозначения используется для представления внутренней структуры интерфейса, которая может быть важна для реализации. Применительно к диаграмме компонентов зависимости могут связывать компоненты и импортируемые этим компонентом интерфейсы, а также различные виды компонентов между собой. В первом случае рисуют стрелку от компонента-клиента к импортируемому интерфейсу (рис. 3). Наличие такой стрелки означает, что компонент не реализует соответствующий интерфейс, а использует его в процессе своего выполнения. Причем на этой же диаграмме может присутствовать и другой компонент, который реализует этот интерфейс. Так, например, изображенный ниже фрагмент диаграммы компонентов представляет информацию о том, что компонент с именем "main.exe" зависит от импортируемого интерфейса IDialog, который, в свою очередь, реализуется компонентом с именем "image.java". Для второго компонента этот же интерфейс является экспортируемым.

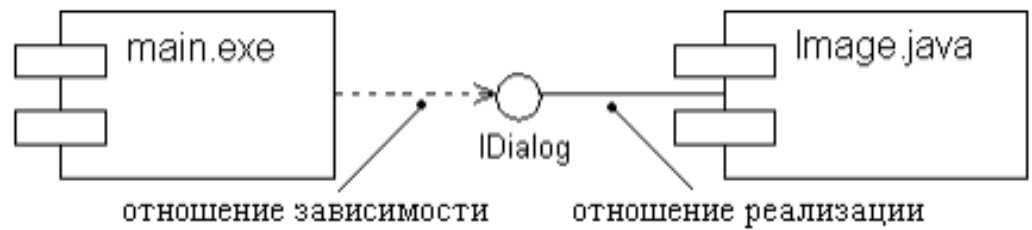


Рисунок 3 - Фрагмент диаграммы компонентов с отношением зависимости

На диаграмме компонентов также могут быть представлены отношения зависимости между компонентами и реализованными в них классами (рис. 4). Эта информация имеет важное значение для обеспечения согласования логического и физического представлений модели системы.

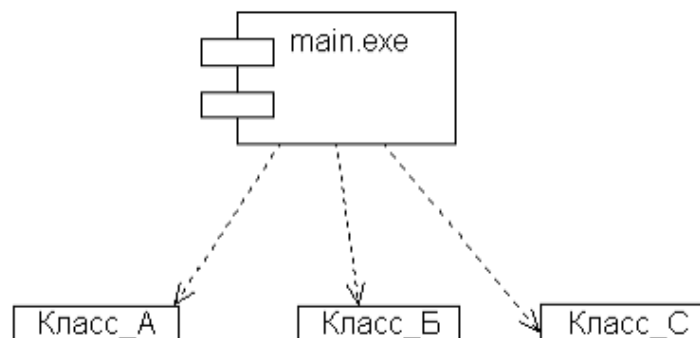


Рисунок 4 - Графическое изображение зависимости между компонентом и классами

Методика и порядок выполнения работы

1. Изучить теоретический материал об интерфейсах в uml, изучить примеры их построения. Рассмотреть материал для построения диаграммы компонентов.
2. Для выполнения практической работы рекомендуется использовать графический редактор Draw (<https://app.diagrams.net/>) или любой другой графический редактор, поддерживающий элементы UML.
3. Необходимо выполнять работу в соответствии с ранее выбранным вариантом (индивидуальный вариант, тема курсовой работы).

4. Построить диаграмму компонентов в соответствии с вариантом.
5. Построить диаграмму интерфейсов в соответствии с вариантом.
6. Оформить отчет с подробным описанием этапов выполнения работы.
7. Ответить на контрольные вопросы письменно в отчете.

Контрольные вопросы

1. Для чего используется диаграмма компонентов UML?
2. На диаграмме компонентов компоненты могут быть каких видов?
3. На диаграмме компонентов отношением зависимости могут связываться
4. Внутри компоненты на диаграмме компонентов могут изображаться
5. Что такое Интерфейсы в UML?

Практическая работа № 13

Разработка концепции рыночного программного продукта

Цель работы и содержание: закрепление теоретических знаний и формирование практических навыков по инициации программного проекта. В процессе выполнения работы студент должен продемонстрировать способность готовить коммерческие предложения по разработке рыночных программных продуктов с различными вариантами реализации

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Задание.

Коллективом программистов на основе накопленного профессионального опыта принято решение об открытии собственного бизнеса. На общем собрании коллектива были определены необходимые для выхода на рынок с новым оригинальным ПП мероприятия: 1.

Проанализировать горизонтальные рынки корпоративных продаж, определить и описать несколько привлекательных идей разработки оригинальных программных продуктов и/или услуг, которые можно предложить клиентам. 2. Провести с использованием метода экспертных оценок выбор наиболее привлекательных идей по разработке ПП. 3. Разработать концепции реализации привлекательных идей в виде рыночных программных продуктов. 4. Провести оценки перспективности концепций с использованием метода экспертных оценок; гибридной модели функциональных зависимостей.

Порядок выполнения лабораторной работы:

1. Создать творческое ядрокоманды в составе 4-х человек по разработке будущего программного продукта.

2. Методом мозгового штурма выбирать 2–3 наиболее перспективных рыночных сегмента. В каждом из сегментов выделить 2–3 привлекательные идеи по разработке оригинальных ПП, определить типы потенциальных потребителей, описать каждую идею.

3. Провести в открытой дискуссии обсуждение идей, кратко описать каждую из них в следующем формате: полное и сокращенное наименование будущего ПП; необходимость или потребность будущего программного продукта в выделенном сегменте рынка, зачем нужен этот продукт и / или услуга, какие проблемы конкретных пользователей он позволит решить; краткое описание программного продукта и/или услуг, которые будут предложены на рынок, наличие конкурентов; ожидаемые длительность и затраты на разработку ПП. Провести ранжирование идей каждым членом команды по степени убыванию их привлекательности. Наиболее привлекательной идее присваивается ранг равный единице, второй — ранг равный двум и т.д. Интегральная оценка привлекательности идеи определяется суммированием рангов экспертов. Отобрать две наиболее привлекательных идеи для дальнейшей проработки.

4. Описать каждую идею в виде концепции программного продукта и / или услуги.

4.1. Необходимость и / или потребность для общества будущего программного продукта. Необходимость и / или потребность для общества будущего программного продукта предлагается рассматривать в четырех аспектах: коммерческая привлекательность (появление новой потребности рынка в программных продуктах или услугах, изменение потребительского спроса на ПП); изменение законодательства, регулирующего состав и содержание бизнес процессов в конкретных предметных областях; научно-технический прогресс в области развития программно-аппаратных средств информатизации (например, проект по разработке программного продукта для

портативных устройств); потребность отраслей социальной сферы в информационно-коммуникационных услугах, финансирование которых берет на себя государство. После обоснования необходимости следует раскрыть следующие вопросы: какие у потенциального заказчика существуют проблемы, насколько значимо для него решение данных проблем; зачем нужен данный продукт, какова его основная идея, что собираетесь разрабатывать и какие требования к ПП могут предъявлять потенциальные пользователи; кому собираетесь предлагать ваш продукт или услугу; какой полезный эффект можно извлечь потенциальный потребитель от использования продукта.

4.2. Цели, ограничения, допущения и основные результаты программного проекта. Цели программного проекта следует определять в виде желаемого результата, достигаемого командой проекта при его успешной реализации. Формулировки целей должны быть конкретными, измеримыми, согласованными, реальными, ограниченными по срокам. При описании желаемого результата программного продукта следует делать акцент на преимуществах, которые этот продукт несет потенциальным покупателям: чем отличается ваш продукт от продуктов конкурентов; обладает ли продукт какими-либо новыми уникальными особенностями; сколько времени уйдет на разработку такого продукта; если в вашем продукте нет ничего особенно выдающегося, то что же в нем может привлечь покупателя. В качестве одного из результатов должно быть краткое описание архитектуры будущего программного продукта, как совокупности программных модулей (компонентов), с перечислением их функционала.

4.3. Основные сегменты рынка и потенциальные пользователи. Разработка данного раздела концепции должна начинаться с выделения его целевых сегментов рынка и выявления множества потенциальных потребителей, которым может быть интересен продукт, далее необходимо описать способы продвижения ПП в выбранных сегментах рынка, выделить перечень услуг, отвечающих запросам потребителей. Кроме этого предлагается привести ответы на следующие вопросы: как вы собираетесь привлечь покупателей; с кем вы собираетесь конкурировать в выбранных сегментах рынка, знаете ли вы производителей аналогичных продуктов; за

какую цену продаются

аналогичные продукты; каким образом вы собираетесь продавать продукт; возможные каналы поставки продукта, как будут организованы возникающие взаимоотношений с пользователями.

4.4. Экономика программного проекта В данном разделе необходимо определить трудозатраты на создание ПП (в человеко-месяцах), бюджет разработки, рыночную цену продажи одной лицензии, срок окупаемости проекта.

4.5. Потенциал исполнителей Кратко описывается история успеха команды проекта, наличие сертификатов и других документов, подтверждающих потенциал, определяется, сколько и каких исполнителей понадобится для реализации проекта, потребуется ли передача отдельных работ на аутсорсинг.

4.6 Ожидаемые риски программного проекта Оценка риска реализации проекта должна ответить на вопрос: есть ли у команды реальная возможность реализовать проект: в приемлемые сроки, функционалом, востребованным потенциальным пользователем, и в пределах имеющихся финансовых ресурсов?

5. Оценить перспективность концепций с использованием метода экспертных оценок

5.1. Выбрать и обосновать состав показателей (критериев) оценки перспективности концепций, для каждого из показателей в интервале {0-1} определить коэффициент относительной важности для достижения целей проекта, более значимому критерию присваивается более высокая оценка.

5.2. Провести оценку перспективности концепций каждым членом команды проекта оценку производить по следующей шкале (табл. 1).

Шкала оценивания

Качественная оценка	Перспективность концепции				
	Слабая	Умеренная	Заметная	Высокая	Очень высоко-
Количественная оценка	[1-2]	[3-4]	[5-6]	[7-8]	[9-10]

5.3. Вычислить интегральную экспертную оценку перспективности каждой концепции по выражению (1), определить степень достоверности экспертизы, обсудить результаты.

(1) 6. Оценить перспективность концепций с использованием гибридной

модели функциональных зависимостей

6.1. Выбирать и обосновать состав первичных показателей (показателей первого уровня) модели, оценить каждую концепцию по выбранной системе показателей.

6.2. Построить сеть функциональных зависимостей. Привести формулы и вычислить значения промежуточных показателей и интегральной оценки перспективности каждой концепции. Обсудить результаты.

7. Провести анализ и обсуждение результаты оценивания перспективности концепций, выбрать наиболее перспективную концепцию.

Практическая работа № 14

Структурная декомпозиция работ и формирование календарного плана. Реализации программного проекта

Цель работы и содержание: закрепление теоретических знаний и формирование практических навыков по выявлению и структуризации множества процессов, работ и заданий программного проекта

Формируемые компетенции: УК-2, УК-3, УК-6, УК-9

Задание.

Команда проекта разработала концепцию будущего программного продукта, нашла источники финансирования и приняла решение о начале работы. Для формирования календарного плана реализации проекта необходимо: – с учетом специфики будущего ПП выбрать наиболее применимую модель жизненного цикла разработки ПП; – провести структурную декомпозицию работ проекта; – разработать календарный план реализации проекта; – оценить риски, связанные с возможным срывом сроков выполнения работ проекта, выбрать стратегию управления риском и предложить мероприятия по его снижению. Порядок выполнения работы

1. Ознакомиться с различными моделями жизненного цикла разработки и методикой выбора модели жизненного цикла разработки ПП.

2. Проанализировать особенности концепции программного проекта, выбрать и описать в соответствии с методикой выбора модели жизненного цикла разработки ПП его основные характеристики.

3. Разработать алгоритм выбора подходящей модели жизненного цикла разработки и выбрать модель. Привести краткое описание выбранной модели, ее достоинства, недостатки, области применения. Принять решение о целесообразности использования данной модели либо обосновать необходимость использования другой.

4. Провести структурную декомпозицию работ программного проекта.

4.1. Описать архитектуру программного продукта в виде взаимосвязей совокупности программных комплексов, программ, программных модулей и интерфейсов.

4.2. В соответствии с рекомендациями стандарта ГОСТ 12207 или IEEE 1074 и особенностями проекта выбрать и описать совокупность фаз, процессов и действий жизненного цикла выбранной модели. Фаза — группы логически связанных процессов (действий), в ходе осуществления которых выполняется вполне конкретная часть проекта. Действие (работа) — выполнение в процессе реализации проекта имеющее ожидаемые продолжительность, стоимость и прогнозируемые потребности в ресурсах. Процесс — ряд действий (работ), выполнение которых приводит к конкретному результату.

4.3. Выбрать для каждого из элементов архитектуры на фазе «Разработка» набор процессов и работ, представить их в виде иерархического вложенного списка работ.

5. Разработать календарный план реализации программного проекта составить скелетный план проекта; определить длительность выполнения каждой работы; установить типы взаимосвязи между работами; выбрать, обосновать и установить необходимые задержки; □ установить крайний срок завершения проекта; сформировать календарный план проекта в виде диаграммы Ганта; сформировать календарный план проекта в виде сетевого графика.

6. Провести анализ работ, лежащих на критическом пути, и оценить вероятности проявления риска невыполнения их в срок; выбрать стратегию управления риском и предложить мероприятия по его снижению.

Список рекомендуемой литературы

Перечень основной литературы

- 1 Управление проектами с использованием Microsoft Project Электронный ресурс : учебное пособие / Т.П. Пухначева / Н.А. Иванчева / Т.С. Васючкова / М.А. Держо. - Управление проектами с использованием Microsoft Project, 2019-12-01. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 147 с
- 2 Мостовой, Я.А. Управление программными проектами Электронный ресурс: учебное пособие / Я.А. Мостовой. - Самара: Поволжский государственный университет телекоммуникаций и информатики, 2016. - 103 с.
- 3 Матвеева, Л. Г. Управление ИТ-проектами : учебное пособие / Л.Г. Матвеева, А.Ю. Никитаева ; Министерство образования и науки РФ ; Южный федеральный университет. - Таганрог : Издательство Южного федерального университета, 2016. - 227 с.
- 4 Рыбалова, Е.А. Управление проектами Электронный ресурс: учебно-методическое пособие / Е.А. Рыбалова. - Томск: Томский государственный университет систем управления и радиоэлектроники, 2015. - 149 с.
- 5 Рыбалова, Е.А. Управление проектами Электронный ресурс: учебное пособие / Е.А. Рыбалова. - Томск: Томский государственный университет систем управления и радиоэлектроники, 2015. - 206 с.
- 6 Ехлаков, Ю. П. Управление программными проектами: учебное пособие / Ю.П. Ехлаков ; Министерство образования и науки Российской Федерации ; Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Эль Контент, 2014. - 140 с.

2. Перечень дополнительной литературы

- 1 Управление проектом. Основы проектного управления : [учебник] / М.Л. Разу, А.М. Лялин, Т.М. Бронникова и др. ; под ред. М.Л. Разу ; Гос. ун-т управления. - 3-е изд., перераб. и доп. - М. : КНОРУС, 2011. - 768 с.
- 2 Управление проектом : основы проектного управления : учебник для вузов / [М.Л. Разу, А.М. Лялин, Т.М. Бронникова, и др.] ; под ред. М.Л. Разу ; Гос. ун-т упр. - 4-е изд., стер. - Москва : КноРус, 2012. - 755 с.
- 3 Ричард, Ньютон. Управление проектами от А до Я Электронный ресурс : практическое пособие / Ньютон Ричард. - Управление проектами от А до Я, 2019-02-15. - Москва : Альпина Паблишер, 2016. - 180 с.

Список рекомендованной литературы

Список основной литературы

- 1 Управление проектами с использованием Microsoft Project Электронный ресурс : учебное пособие / Т.П. Пухначева / Н.А. Иванчева / Т.С. Васючкова / М.А. Держо. - Управление проектами с использованием Microsoft Project, 2019-12-01. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 147 с
- 2 Мостовой, Я.А. Управление программными проектами Электронный ресурс: учебное пособие / Я.А. Мостовой. - Самара: Поволжский государственный университет телекоммуникаций и информатики, 2016. - 103 с.
- 3 Матвеева, Л. Г. Управление ИТ-проектами : учебное пособие / Л.Г. Матвеева, А.Ю. Никитаева ; Министерство образования и науки РФ ; Южный федеральный университет. - Таганрог : Издательство Южного федерального университета, 2016. - 227 с.
- 4 Рыбалова, Е.А. Управление проектами Электронный ресурс: учебно-методическое пособие / Е.А. Рыбалова. - Томск: Томский государственный университет систем управления и радиоэлектроники, 2015. - 149 с.
- 5 Рыбалова, Е.А. Управление проектами Электронный ресурс: учебное пособие / Е.А. Рыбалова. - Томск: Томский государственный университет систем управления и радиоэлектроники, 2015. - 206 с.
- 6 Ехлаков, Ю. П. Управление программными проектами: учебное пособие / Ю.П. Ехлаков ; Министерство образования и науки Российской Федерации ; Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Эль Контент, 2014. - 140 с.

Список дополнительной литературы:

- 1 Управление проектом. Основы проектного управления : [учебник] / М.Л. Разу, А.М. Лялин, Т.М. Бронникова и др. ; под ред. М.Л. Разу ; Гос. ун-т управления. - 3-е изд., перераб. и доп. - М. : КНОРУС, 2011. - 768 с.
- 2 Управление проектом : основы проектного управления : учебник для вузов / [М.Л. Разу, А.М. Лялин, Т.М. Бронникова, и др.] ; под ред. М.Л. Разу ; Гос. ун-т упр. - 4-е изд., стер. - Москва : КноРус, 2012. - 755 с.
- 3 Ричард, Ньютон. Управление проектами от А до Я Электронный ресурс : практическое пособие / Ньютон Ричард. - Управление проектами от А до Я, 2019-02-15. - Москва : Альпина Паблишер, 2016. - 180 с.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»,

необходимых для освоения дисциплины:

1. Википедия. Свободная Интернет-энциклопедия (ru.wikipedia.org)
2. Федеральный портал Российское образование (edu.ru)
3. Союз образовательных сайтов (allbest.ru)
4. Официальный сайт Министерства образования и науки Российской Федерации (mon.gov.ru)
6. Интернет-университет информационных технологий (intuit.ru)

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное
автономное образовательное
учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К САМОСТОЯТЕЛЬНОЙ РАБОТЕ

«Проектная деятельность»

Направление подготовки 09.03.04 «Программная инженерия» Направленность
(профиль) «Прикладное программирование и интернет-технологии»
Квалификация выпускника: бакалавр

Ставрополь

2026

ВВЕДЕНИЕ

Цель изучения дисциплины «Проектная деятельность» формирование системного методического подхода к проектной деятельности и приобретение практических навыков проектной работы IT-области, формирование высокой проектной культуры, а также набора общекультурных и профессиональных компетенций будущего бакалавра по направлению подготовки 09.03.02 Информационные системы и технологии. Задачами обучения являются:

- усвоение роли грамотной организации проектной деятельности для эффективного решения проектных задач различной сложности;
- изучение основ и методов планирования этапов будущего проекта;
- изучение основ управления в проектной деятельности;
- обретение навыков формирования и формулирования задач для индивидуальной и совместной (коллективной) проектной деятельности;
- применение инновационных креативных технологий и методик для создания и совершенствования творческих идей;
- обретение навыков правильного оформления готового проекта для презентации (в том числе, заказчику), для выставки, просмотра, печати, архива.

Дисциплина «Проектная деятельность» относится к циклу Б1 вариативная часть, дисциплины по выбору (Б1.О.07). Ее освоение происходит в 1 и 2 семестре. Для успешного освоения дисциплины «Проектная деятельность» необходимы знания, полученные в ходе освоения дисциплин:

- Цифровая грамотность и обработка больших данных
- Введение в специальность.

Знания, полученные в результате освоения дисциплины «Проектная деятельность» могут быть использованы при подготовке отчетов по преддипломной практике и выпускной квалификационной работы, а также в освоении дисциплины

- Проектная деятельность.

В ходе изучения дисциплины «Проектная деятельность» у обучающегося формируются набор следующих компетенций:

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
УК-2 – способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2_{ид-1.1} - формулирует цель проекта, определяет совокупность взаимосвязанных задач, обеспечивающих ее достижение и определяет ожидаемые результаты решения задач. УК-2_{ид-1.2} - разрабатывает план действий для решения задач проекта, выбирая оптимальный способ их решения, исходя из действующих правовых норм и имеющихся ресурсов и ограничений УК-2_{ид-1.3} - обеспечивает выполнение проекта в соответствии с установленными целями, сроками и затратами, исходя из действующих правовых норм, имеющихся ресурсов и ограничений, в том числе с использованием цифровых инструментов	Результаты демонстрации знаний о грамотно сформулированной цели проекта. Верно определена совокупность взаимосвязанных задач, обеспечивающих ее достижение и определяет ожидаемые результаты решения задач. Результаты применения современных подходов для разработки плана действий для решения задач проекта с помощью оптимальных способов решения, исходя из действующих правовых норм и имеющихся ресурсов и ограничений являются верными. Результаты выполнения проекта в соответствии с установленными целями, сроками и затратами, исходя из действующих правовых норм, имеющихся ресурсов и ограничений, в том числе с использованием цифровых инструментов.

УК-3 – способен осуществлять социальное взаимодействие и реализовывать свою роль в команде	УК-3_{ид-3.1} участвует в межличностном и групповом взаимодействии, используя инклюзивный подход, эффективную коммуникацию, методы командообразования и командного взаимодействия при совместной работе в рамках поставленной задачи. УК-3_{ид-3.2} – обеспечивает работу команды для получения оптимальных результатов совместной работы, с учетом индивидуальных возможностей её членов, использования методологии достижения успеха, методов, информационных технологий и технологий форсайта. УК-3_{ид-3.3} – обеспечивает выполнение поставленных задач на основе мониторинга командной работы и своевременного реагирования на существенные отклонения	Проводит предпроектное обследование предметной области, выявляет бизнес-проблему для разработки проекта. На основе предпроектного обследования формирует цели и задачи проекта, формирует концепцию проекта Разрабатывает техническое задание на основе ГОСТ разрабатывает план реализации проекта и его этапов, формирует требования к разработке, устанавливает виды работ и их сроки. Использует CASE-системы для проектирования, управления проектом и его ресурсами, составляет график работ, использует методы предпроектного обследования и проектирования.
УК-6 – способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни	УК-6_{ид-6.1} – устанавливает личные и профессиональные цели в соответствии с уровнем своих ресурсов и приоритетов действий, для успешного развития в избранной сфере профессиональной деятельности УК-6_{ид-6.2} – реализует и корректирует стратегию личностного и профессионального развития, с учетом условий, средств, личностных возможностей, этапов карьерного роста, временной перспективы развития деятельности и требований рынка труда УК-6_{ид-6.3} – критически оценивает эффективность использования времени и других ресурсов при решении поставленных задач в избранной сфере профессиональной деятельности	Анализирует и оценивает риски при проведении проектов. Определяет цели и задачи проекта, формирует его концепцию и план. Планирует и контролирует выполнение работ по проекту.
УК-9 – Способен принимать обоснованные экономические решения в различных областях жизнедеятельности	УК-9_{ид-9.1} - понимает базовые принципы функционирования экономики и экономического развития, цели и формы участия государства в экономике УК-9_{ид-9.2} – применяет методы личного экономического и финансового планирования для достижения текущих и долгосрочных финансовых целей УК-9_{ид-9.3} – использует финансовые инструменты для управления личными	Управляет бюджетом и ресурсами проекта. Разрабатывает стратегии управления изменениями в рамках проектов. Анализирует и оценивает влияние социальных, экономических и экологических факторов на проектную деятельность

	финансами, контролирует собственные экономические и финансовые риски	
--	---	--

1. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Самостоятельная работа студентов является важнейшим условием формирования научного способа познания. Она проводится накануне каждого лабораторного занятия и включает изучение необходимого для выполнения лабораторных работ и подготовки к практическим занятиям теоретического материала, а также подготовки отчетов по выполненным лабораторным и практическим занятиям. Подготовленный материал оформляется в виде отчета. Самостоятельные занятия (СЗ) являются одной из активных форм обучения. Самостоятельные занятия по дисциплине

«Проектная деятельность» имеют целью: - закрепить и углубить знания, полученные студентами на лекциях и в процессе лабораторных и практических занятий; - привить практические навыки в оформлении организационных, распорядительных и информационно-справочных документов. Самостоятельные занятия проводятся в специализированных аудиториях, оборудованных средствами вычислительной техники и возможностью пользования Internet-ресурсами, в библиотеке. Предлагаемые методические рекомендации содержат информацию для студентов, необходимую при подготовке и проведении лабораторных занятий по дисциплине «Основам проектной деятельности».

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Самостоятельная работа по дисциплине «Основы проектной

деятельности» состоит из двух частей:

1. Самостоятельное изучение вопросов;
2. Самостоятельная разработка проекта на базе умений и навыков, полученных на лабораторных работах.

3. СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

3.1. Темы для самостоятельного изучения

1. Методика выбора модели жизненного цикла разработки ПП
2. ГОСТ 12207. Процессы жизненного цикла программного обеспечения. Состав работ и задач процесса «Разработка» ПП
3. Процессы и действия, выполняемые в рамках ЖЦ разработки ПП (документ IEEE 1074)
4. ГОСТ 19.102-77 ЕСПД «Стадии разработки

В процессе самостоятельного изучения вопросов курса студент изучает предложенные им по рекомендуемому преподавателем списку литературы и предоставляет преподавателю краткий конспект, по которому в дальнейшем проводится индивидуальное собеседование.

На зачетной неделе студент должен предоставить готовый проект.

4. ОРГАНИЗАЦИЯ КОНТРОЛЯ ЗНАНИЙ, ПОЛУЧЕННЫХ ПРИ ВЫПОЛНЕНИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

В процессе собеседования и защиты самостоятельного проекта:

Оценка «отлично» выставляется студенту, если теоретическое содержание вопросов самостоятельного изучения освоено полностью, без

пробелов; исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует в ответе дополнительный материал все предусмотренные программой задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному; анализирует полученные результаты; проявляет самостоятельность при выполнении заданий.

Оценка «хорошо» выставляется студенту, если теоретическое и практическое содержание вопросов самостоятельной работы освоено полностью, необходимые практические компетенции в основном сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос.

Оценка «удовлетворительно» выставляется студенту, если теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, большинство предусмотренных программой заданий выполнено, но в них имеются ошибки, при ответе на поставленный вопрос студент допускает неточности, недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении программного материала. Практические навыки освоены частично.

Оценка «неудовлетворительно» выставляется студенту, если он не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические работы, необходимые практические компетенции не сформированы, большинство предусмотренных программой обучения учебных заданий не выполнено, качество их выполнения оценено числом баллов, близким к минимальному.

5. РЕКОМЕНДАЦИИ ПО РАБОТЕ С ЛИТЕРАТУРОЙ И ИСТОЧНИКАМИ

5.1 Рекомендации студентам по организации работы с литературой и источниками Изучение литературы и источников необходимо начинать с прочтения соответствующих глав учебных изданий, учебных пособий или литературы, рекомендованной в качестве основной или дополнительной по дисциплине «Проектная деятельность», которые прямо или косвенно относятся к отрабатываемой теме. Изучая литературу и источники, студенту рекомендуется вести краткий конспект. Однако не следует переписывать все содержание изучаемой темы, нужно выписывать лишь основные идеи и главные на ваш взгляд мысли. В отдельных случаях, когда встречаются важные определения, понятия, необходимый фактический материал и примеры, статистическая информация, имеющие отношение к изучаемой теме, студенту следует выписать их в виде цитат с полным указанием библиографических источников. Конспектирование рекомендуемой литературы и источников необходимо вести с распределением собранных материалов по отдельным главам и параграфам согласно учебно-тематическому плану. Необходимо выписывать все выходные данные по используемой литературе и источникам. Важным этапом при работе с рекомендуемой литературой и источниками является изучение законодательных и нормативных актов федерального, регионального, местного и ведомственного уровней. При изучении Указов Президента РФ, Законов и Кодексов РФ, постановлений, положений, рекомендаций и т.д., студент должен выяснить все изменения и дополнения, которые могли быть внесены после их выхода в свет. Основой технологии интенсификации обучения на платформе цифровых образовательных технологий являются учебно-иллюстрационные материалы (опорный конспект) по дисциплине «Проектная деятельность».

Список рекомендованной литературы

Список основной литературы

- 1 Управление проектами с использованием Microsoft Project Электронный ресурс : учебное пособие / Т.П. Пухначева / Н.А. Иванчева / Т.С. Васючкова / М.А. Держо. - Управление проектами с использованием Microsoft Project, 2019-12-01. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 147 с
- 2 Мостовой, Я.А. Управление программными проектами Электронный ресурс: учебное пособие / Я.А. Мостовой. - Самара: Поволжский государственный университет телекоммуникаций и информатики, 2016. - 103 с.
- 3 Матвеева, Л. Г. Управление ИТ-проектами : учебное пособие / Л.Г. Матвеева, А.Ю. Никитаева ; Министерство образования и науки РФ ; Южный федеральный университет. - Таганрог : Издательство Южного федерального университета, 2016. - 227 с.
- 4 Рыбалова, Е.А. Управление проектами Электронный ресурс: учебно-методическое пособие / Е.А. Рыбалова. - Томск: Томский государственный университет систем управления и радиоэлектроники, 2015. - 149 с.
- 5 Рыбалова, Е.А. Управление проектами Электронный ресурс: учебное пособие / Е.А. Рыбалова. - Томск: Томский государственный университет систем управления и радиоэлектроники, 2015. - 206 с.
- 6 Ехлаков, Ю. П. Управление программными проектами: учебное пособие / Ю.П. Ехлаков ; Министерство образования и науки Российской Федерации ; Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Эль Контент, 2014. - 140 с.

Список дополнительной литературы:

- 1 Управление проектом. Основы проектного управления : [учебник] / М.Л. Разу, А.М. Лялин, Т.М. Бронникова и др. ; под ред. М.Л. Разу ; Гос. ун-т управления. - 3-е изд., перераб. и доп. - М. : КНОРУС, 2011. - 768 с.
- 2 Управление проектом : основы проектного управления : учебник для вузов / [М.Л. Разу, А.М. Лялин, Т.М. Бронникова, и др.] ; под ред. М.Л. Разу ; Гос. ун-т упр. - 4-е изд., стер. - Москва : КноРус, 2012. - 755 с.
- 3 Ричард, Ньютон. Управление проектами от А до Я Электронный ресурс : практическое пособие / Ньютон Ричард. - Управление проектами от А до Я, 2019-02-15. - Москва : Альпина Пабlishер, 2016. - 180 с.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»,

необходимых для освоения дисциплины:

1. Википедия. Свободная Интернет-энциклопедия (ru.wikipedia.org)

2. Федеральный портал Российское образование (edu.ru)
3. Союз образовательных сайтов (allbest.ru)
4. Официальный сайт Министерства образования и науки Российской Федерации (mon.gov.ru)
6. Интернет-университет информационных технологий (intuit.ru)