

Введение

Для выполнения практических заданий необходимо в первую очередь ознакомиться с общими указаниями по выполнению работ по дисциплине «Проектирование информационных систем».

Практикум выполняется в форме последовательного выполнения проектных заданий на основе общего задания, что позволит сформировать комплект проектной документации для создания автоматизированной информационной системы (ИС) определённого назначения. В дальнейшем этот комплект проектной документации используется для создания компонентов информационного и программного обеспечения, а также проектной и эксплуатационной документации ИС.

Задания индивидуально. В отдельных случаях (при создании сложных систем или при создании информационной системы по заданию кафедры) после согласования с преподавателем возможно выполнение работ проектными группами в составе двух человек.

Для выполнения проектных заданий в качестве нормативных документов используются государственные стандарты РФ [1–2], руководящие документы и методические рекомендации по моделированию объектов предметной области [7–8].

Преподаватель исполняет роль представителя заказчика (предприятия, которое заключает договор на выполнение проектных работ по созданию ИС) и роль руководителя проектов. В роли представителя заказчика преподаватель консультирует по вопросам постановки проектных задач (с ним проектные группы согласуют вопросы, касающиеся содержания и возможных изменений проектных заданий), а в качестве руководителя проектов преподаватель консультирует по вопросам выполнения проектных задач, проверяет и оценивает результаты работы проектных групп (выполнение графика, состава и качества решения

проектных задач), заслушивает доклады о промежуточных результатах. Следует иметь в виду, что начальные данные для проектирования являются неполными и неформализованными, поэтому целый ряд вопросов (особенно на начальных стадиях проектирования) требует уточнения, консультаций у представителя заказчика (состав и содержание производственных функций, состав и содержание информационных потоков, возможные значения реквизитов документов и ряд других) и руководителя проектов (по методике выполнения проектных заданий и принятия проектных решений).

Для подготовки комплекта проектной документации по созданию будущей автоматизированной информационной системы студенты должны выполнить в течение **пять** проектных заданий. В результате выполнения каждого промежуточного проектного задания должен быть сформирован отчёт в письменной форме и в виде текстового файла (предпочтительно в формате .docx). Отчет должен быть предъявлен для проверки руководителю проекта (преподавателю). Если по отчёту сделаны замечания, то исполнителям (проектным группам) следует внести исправления в отчёт. По результатам работы подготовить краткий доклад с презентацией для выступления на очередном занятии. Результат выполнения каждого проектного задания оценивается; по следующим факторам: выполнение графика работ, качество оформления отчёта и качество устного доклада с презентацией (рисунки 2–3). На титульном листе отчета преподаватель ставит оценки за каждый оцениваемый фактор (см. выше). Дополнительные баллы выставляются только после того, как преподаватель зачтет все работы, выполненные студентом.

Максимальное количество баллов за активность по дисциплине равно 25 баллов. Для получения баллов за активность по дисциплине в течение учебного семестра необходимо выполнить следующие действия с соблюдением следующих требований:

Действие	Мин балл	Макс балл	Срок выполнения	Критерии оценки
Своевременная сдача всех практических работ	0	12,5	25.12.2020	Глубина проработки материала, раскрытие темы, умения и навыки работы с нормативными документами, понимание методологии. Грамотное техническое описание. Оформление пояснительной записки
Качественное оформление работы согласно требованиям ГОСТ 7.32–2017, СМКО МИРЭА и ГОСТ Р 7.0.100–2018	0	12,5	25.12.2020	Соответствие требованиям нормативных документов

Правила начисления дополнительных баллов за активность

Дополнительные баллы за активность начисляются следующим образом:

В семестре предусмотрено выполнение 5 практических работ. Каждая работа оценивается по 2 параметрам

Технологическая реализация (смысловое и научное наполнение консида работы) максимум в 20 баллов, при этом:

20 баллов – работа выполнена безукоризненно: все задачи решены, цель работы достигнута.

15 баллов – работа выполнена хорошо: цель работы достигнута, решены все задачи, хотя и не полностью.

10 баллов – работа в принципе выполнена (между хорошо и удовлетворительно): цель можно считать достигнутой при том, что-либо одна

задача не решена, либо все задачи решены, но не полностью.
5 баллов – работа выполнена частично. Цель достигнута частично, не решено более одной из поставленных задач.
0 баллов – работа не выполнена.

Возможно выставление промежуточных баллов, для более объективной оценки работы.

Оформление и сроки сдачи каждой работы также оцениваются в 20 баллов при этом до 10 баллов присваивается за своевременную сдачу работы и 10 баллов за оформление. Оформление работы должно соответствовать требованиям ГОСТ 7.32-2017 и СМКО МИРЭА.

20 баллов – работа выполнена в срок, оформление полностью соответствует ГОСТ 7.32-2017 и СМКО МИРЭА, смысл работы передан предельно ясно;
15 баллов – работа сдана с опозданием не более 3 рабочих дней, а оформление минимально отличается от ГОСТ 7.32-2017 и СМКО МИРЭА, смысл работы передан ясно;
10 баллов – работа сдана с опозданием более 3, но не более 6 рабочих дней, а оформление минимально отличается от ГОСТ 7.32-2017 и СМКО МИРЭА, смысл работы передан ясно;
5 баллов – работа сдана с опозданием более 6, но не более 9 рабочих дней, оформление минимально отличается от ГОСТ 7.32-2017 и СМКО МИРЭА, смысл работы передан ясно;
0 баллов – работа сдана с опозданием в более 9 рабочих дней и более, но не более 9 рабочих дней, оформление минимально отличается от ГОСТ 7.32-2017 и СМКО МИРЭА, смысл работы передан правильно.

Возможно выставление промежуточных баллов, для более объективной оценки работы. При этом работы, не соответствующие ГОСТ 7.32-2017 и СМКО МИРЭА, смысл которых не ясен возвращаются на переделку авторам.

Таким образом студент может получить 100 баллов за технологическую часть выполнения всех работ и 100 баллов за своевременное выполнение и корректное оформление всех работ. Баллы умножаются на коэффициент 0,125. Таким образом за активность в системе СДО МИРЭА можно получить дополнительно 12,5 баллов за технологическую часть выполнения всех работ и 12,5 баллов за своевременно корректное оформление всех работ, что в сумме соответствует максимально возможным 25 баллам.

Работа 1 формирование требований к системе

Этап разработки требований – один из наиболее ответственных этапов создания программного продукта. На данной стадии формулируются основные требования к разрабатываемому программному обеспечению. От того, насколько полно определены функции и требования, насколько правильно приняты принципиальные решения, определяющие процесс проектирования, во многом зависит стоимость разработки и ее качество.

Задача формирования требований к ИС является одной из трудно формализуемых, наиболее дорогих и тяжелых для исправления в случае ошибки. Современные инструментальные средства и программные продукты позволяют достаточно быстро создавать ИС по готовым требованиям. Но зачастую эти системы не удовлетворяют заказчиков, требуют многочисленных доработок, что приводит к резкому удорожанию фактической стоимости ИС. Основной причиной такого положения является неправильное, неточное или неполное определение требований к ИС на начальном этапе ЖЦ.

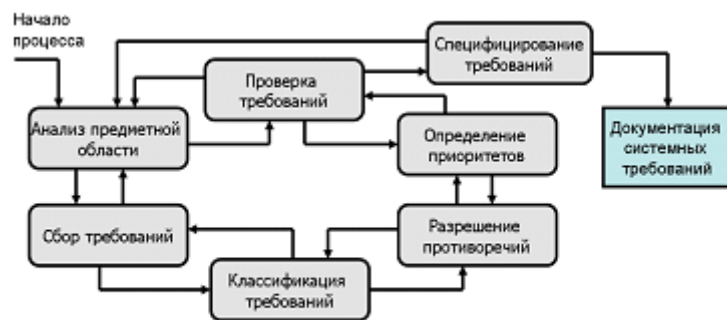


Рисунок 1. Процесс формирования и анализа требований

Этап разработки требований состоит из следующих шагов (рис. 1):

- анализ предметной области, в рамках которой будет эксплуатироваться система;
- сбор требований. Это процесс взаимодействия с лицами, формирующими требования;
- классификация требований. На этом этапе бесформенный набор требований преобразуется в логически связанные группы требований.
- разрешение противоречий. Требования лиц, занятых в процессе формирования требований, могут быть противоречивыми. На данном шаге определяются и разрешаются противоречия такого рода;
- назначение приоритетов. Определяются наиболее важные требования, реализовать которые необходимо в первой версии программы;
- проверка требований на полноту, последовательность и непротиворечивость.

Ошибки, допущенные на стадии сбора требований, составляют от 40 до 60% всех дефектов проекта (Davis, 1993; Leffing well, 1997). Frederick Brooks выразительно определил критическую роль требований при разработке программного обеспечения (ПО) в классическом эссе (1987) «No Silver Bullet:

Essence and Accidents of Software Engineering»: «Строжайшее и единственное правило построения систем ПО – решить точно, что же строить. Никакая другая часть концептуальной работы не является такой трудной, как выяснение деталей технических требований, в том числе и взаимодействие с людьми, с механизмами и с иными системами. Никакая другая часть работы так не портит результат, если она выполнена плохо. Ошибки никакого другого этапа работы не исправляются так трудно».

Разработчики ИС при формировании требований могут столкнуться со следующими проблемами, приводящими к пересмотру списка требований и изменения в ИС [46]:

Недостаточное вовлечение пользователей на всех этапах разработки требований. Для решения данной проблемы необходимо: определить различные классы пользователей ИС; определить основные источники получения информации о потребностях клиентов; выбрать представителей каждого класса пользователей и прочих групп заинтересованных лиц и поработать с ними; согласовать с пользователями ответственных за принятие решений по проекту. В качестве дополнительных классов пользователей можно рассматривать сторонние приложения и аппаратные компоненты, с которыми взаимодействует система. Например, ИС «Деканат», использующая информацию об индивидуальном образовательном маршруте студентов, представляет собой пользовательский класс ИС «Индивидуальный образовательный маршрут». Требования к ИС «Индивидуальный образовательный маршрут» необходимо уточнить у разработчиков ИС «Деканат».

«Разрастание» требований пользователей. Так как требования тщательно прорабатываются и их объем со временем увеличивается, проект часто выходит за установленные рамки, как по срокам, так и по бюджету. Изменения списка требований часто приводит к добавлению программного кода, что может стать причиной нарушения принципов дизайна и разрыва

логических связей внутри ИС. Чтобы минимизировать возможность потери качества ПО, необходимо проанализировать, как возможные изменения отразятся на архитектуре и дизайне, а затем реализовать их непосредственно в программном коде.

Двусмысленность требований. Пользователи ИС могут интерпретировать одно и то же положение по-разному, у нескольких читателей требований возникает разное представление о продукте. Кроме того, двусмысленность зачастую является следствием неточности и плохой детализации требований, в результате чего разработчикам приходится заполнять возникающие пробелы собственными силами. Один из способов обнаружить двусмысленности – пригласить различных представителей классов пользователей для официальной экспертизы требований.

Добавление в ИС функций, ненужных заказчику. Применение диаграммы вариантов использования для извлечения требований поможет сосредоточиться на выборе тех функций и элементов графического интерфейса, которые помогут пользователям выполнять их задачи.

Среди универсальных методов, пригодных для обследования всех уровней управления организации, наиболее важными являются следующие методы:

- изучение документов. Полное представление об объекте автоматизации достигается изучением законодательных актов, приказов, положений как об организации в целом, так и о её структурных подразделениях, и штатного расписания. Документы, описывающие уже работающую или конкурирующую ИС, могут содержать корпоративные или отраслевые стандарты, которых необходимо придерживаться. Опубликованные сравнительные обзоры ИС позволяют выявить недостатки других аналогичных систем, на которые стоит вовремя обратить внимание. Так же аналитикам необходимо изучить учетные формы, перечни и сведения, дающие материал о технологии обработки документов в учреждении;

- наблюдение заключается в непосредственном присутствии на рабочих местах и визуальном фиксировании происходящих процессов, дает возможность разработчикам собрать необходимый исходный материал, проверить и уточнить данные, полученные другими методами. Наблюдение помогает обнаружить неявные требования к системе, которые отражают реальные аспекты её эксплуатации;
- опрос исполнителей (метод интервью) позволяет проверить, уточнить и дополнить данные, полученные при исследовании документов и наблюдении деятельности предприятия, и проводится вместе с ними. Данный метод помогает выявить распределение обязанностей между сотрудниками, виды работ, выполняемых каждым сотрудником подразделения;
- личное участие аналитиков в выполнении рабочих обязанностей предполагаемых пользователей в организации-заказчике.

Пример администрирования требований к проектируемой информационной системе.

Вся документация на информационную систему состоит из следующих разделов:

- Общая часть
 - Список терминов и определений
 - Описание бизнес-ролей
- Требования
 - Бизнес-требования
 - Общие сценарии
 - Сценарии использования
 - Алгоритмы и проверки
- Системные требования
- Нефункциональные требования
- Требования к интеграции
- Требования к пользовательскому интерфейсу
 - Реализация

- Тестирование
- Руководства
- Управление

Общая часть состоит из двух разделов: списка терминов и их определений и описания бизнес-ролей пользователей. Любая документация по системе, включая, например, тестовые сценарии, опирается на определения, данные в этой части.

Бизнес-требования описывают то, что необходимо бизнес-пользователям. Например, им вовсе не нужен объект системы Пользователь, но зато им нужно иметь возможность поменять стоимость товара в счете и распечатать его. Бизнес-требования состоят из общих сценариев, сценариев использования (use cases) и описания алгоритмов обработки данных. Подробно о разработке подобного рода требований можно узнать из книги Карла И. Вигерса и Джоя Битти [Разработка требований к программному обеспечению](#).

Системные требования описывают свойства и методы всех объектов системы.

Нефункциональных требований будут описаны далее. В процессе проектирования функциональной модели. Отметим, однако, что более подробно с процессом формирования нефункциональных требований, можно ознакомиться в книге [Architecting Enterprise Solutions](#) авторов Paul Dyson, Andrew Longshaw.

Требования к интеграции описывают низкоуровневый интерфейс взаимодействия новой системы с несколькими другими системами компании. В данном пособии мы их рассматривать не будем.

Требования к пользовательскому интерфейсу – отдельная большая тема, которая рассматривается в других учебных курсах.

Давайте рассмотрим подробнее, что такое список терминов и зачем он нужен.

Список терминов и определений

Очень часто при обсуждении функциональности системы разговор заходит в тупик. Еще хуже, если стороны расходятся, думая, что обо всем договорились, но в результате имеют разное понимание того, что надо сделать. Это происходит не в последней степени из-за того, что изначально участники проекта не смогли договориться о том, что значат те или иные термины. Бывает, что даже самые простые слова вызывают проблемы: что такое пользователь, чем отличается группа от роли, кто является клиентом. Поэтому в отличие от описания бизнес-ролей для терминов необходимо давать как можно более точные определения.

Поясню это на примере термина – **Пользователь**. Википедия дает такое определение:

Пользователь — лицо или организация, которое использует действующую систему для выполнения конкретной функции [ссылка на Википедию].

Но нас оно не устраивало по нескольким причинам. Во-первых, в систему может зайти только человек, но не организация. Во-вторых, для нашей системы некорректно настоящее время глагола «использует» — система хранит данные о неактивных или удаленных пользователях, т.е. о тех, которые использовали систему ранее, но не могут в настоящее время. И наконец, у нас есть данные о потенциальных пользователях. Например, мы регистрируем сотрудника компании-клиента, который в дальнейшем может получить (а может и не получить) доступ в систему. Наше определение:

Пользователь — человек, который имеет, имел, или, возможно, будет иметь доступ в систему для совершения операций. Теперь программист, прочитав определение, сразу поймет, почему свойство **Логин** в объекте Пользователь не обязательное.

Термины связаны друг с другом. В термине Пользователь используется «операция», поэтому приведу и ее определение:

Операция — совокупность действий, составляющих содержание одного акта бизнес-деятельности. Операция должна соответствовать требованиям ACID (Atomicity, Consistency, Isolation, Durability). Совокупность операций одного модуля представляет интерфейс взаимодействия клиент-сервер этого модуля.

Как видите, это определение очень важно для всей системы — оно не только связывает пользователя и его бизнес-действия с тем, что должно быть реализовано, но и накладывает требования на то, КАК должна быть реализована система (это КАК было определено ранее при разработке архитектуры) — бизнес-действия внутри операции должны быть внутри транзакции.

Работа над списком терминов происходила постоянно. Мы поддерживали его полноту, т.е. старались, чтобы в документации не было термина, который бы не был определен в этом списке. Кроме того, были случаи, когда мы меняли термины. Например, по прошествии нескольких месяцев с начала написания требований мы решили заменить Контрагент на Компания. Причина была проста: оказалось, что никто не в состоянии в речи, при разговоре, использовать слово «контрагент». А если так, то он должен был быть заменен на что-то более благозвучное.

Часто бывали случаи, когда приходилось прерывать обсуждение и лезть в требования, чтобы понять, подходит ли обсуждаемая функциональность под существующие определения. И для того, чтобы поддержать непротиворечивость требований, мы в итоге должны были или изменять реализацию, или корректировать описания терминов.

В итоге в списке у нас оказалось порядка 200 бизнес- и системных определений, которые мы использовали не только во всей документации, включая, например, и технический дизайн, разрабатываемый

программистами, но и в разговоре, при устном обсуждении функциональности системы.

Второй частью, на которую опиралась вся документация, было описание бизнес-ролей.

Описание бизнес-ролей

Все знают, что используют систему пользователи. Но даже в небольшой системе они обладают разными правами и/или ролями. Наверное, самое простое деление – это администратор и рядовой пользователь. В большой системе ролей может быть несколько десятков и аналитику необходимо заранее об этом подумать и указывать роли при описании общих сценариев (смотри ниже) и в заголовках сценариев использования. Список бизнес-ролей используется для реализации групп и ролей пользователей, назначения им функциональных прав, он необходим тестировщикам, чтобы тестировать сценарии под нужными ролями.

Бизнес-роли пользователей нам не пришлось выдумывать, поскольку в компании были устоявшиеся отделы, роли, функции. Описание ролей было дано на качественном уровне на основе анализа основных функций сотрудников. Окончательное наделение ролей конкретными правами происходило ближе к концу разработки, когда набор функциональных прав стал устойчивым.

Примеры приведены в таблице 1:

Таблица 1. Описание бизнес-ролей пользователей

Пользователи, бизнес-роли

Бизнес-роль	Краткое название	Отдел	Функции
Дилер-менеджер	ДМ	Отдел продаж	Занимается взаимодействием с клиентами по процессам продажи товара. Размещает запросы клиентов, создает счета и грузы в системе. За каждым ДМ закреплен определенный набор клиентов.
Специалист клиентского отдела	СКО	Клиентский отдел	Занимается взаимодействием с клиентами и дилер-менеджерами по вопросам выдачи груза (время отгрузки/доставки, неполный подбор и т.п.), оформлением первичных документов на товары в грузах, оформлением транспортных документов на доставку груза.

Уровни требований

Одной из важных концепций, которую мы применяли при разработке требований, было разделение их на уровни. Алистер Коберн в книге [Современные методы описания функциональных требований к системам](#) выделяет 5 уровней. Мы использовали 4 – три уровня бизнес-требований плюс системные требования:

Бизнес-требования

1. Общие сценарии (соответствует уровню очень белого у Коберна)
2. Сценарии использования (соответствует голубому)
3. Алгоритмы и проверки (скорее черный)

4. Системные требования (нет прямого аналога, скорее черный)

Кроме того наши требования представляли из себя дерево (с циклами). Т.е. общие сценарии уточнялись сценариями использования, которые, в свою очередь, имели ссылки на проверки и алгоритмы. Поскольку мы использовали wiki, физическая реализация такой структуры не представляла проблем. Сценарии использования, алгоритмы и проверки использовали объекты, их свойства и методы, описанные на системном уровне.

Такая методология позволяла нам с одной стороны описывать текущий сценарий настолько подробно, насколько нужно на данном уровне, вынося детали на нижний уровень. С другой стороны, находясь на любом уровне можно было подняться выше, чтобы понять контекст его выполнения. Это так же обеспечивалось функциональностью wiki: сценарии и алгоритмы были написаны на отдельных страницах, а wiki позволяла посмотреть, какие страницы ссылаются на текущую. Если алгоритм использовался в нескольких сценариях, то он в обязательном порядке выносился на отдельную страницу.

Такие фрагменты программисты обычно реализовывали в виде отдельных методов.

Ниже представлена часть нашей иерархии (о содержании речь пойдет дальше).

- Список терминов
- Пользователи, бизнес-роли
- ▼ Требования
 - ▼ Сценарии использования
 - Общие сценарии
 - ▼ Печать документов
 - ▼ Пакетная печать документов на груз
 - Алгоритм перехода ЗнП в статус 'В работе'
 - ▼ Системные требования
 - Задание на печать документов

Важно отметить, что если системный уровень описывал все без исключения объекты системы, то сценарии были написаны далеко не для всех случаев поведения пользователя. Ведь многие объекты, по сути, являлись справочниками, и требования к ним более-менее очевидны и похожи. Таким образом мы сэкономили время аналитика.

Интересен вопрос, кому в проектной команде какой из уровней нужен. Будущие пользователи могут читать общие сценарии. Но уже сценарии использования для них сложны, поэтому аналитик обычно обсуждает сценарии с пользователями, но не отдает их им для самостоятельного изучения. Программистам обычно нужны алгоритмы, проверки и системные требования. Вы однозначно можете уважать программиста, который читает сценарии использования. Тестировщикам (как и аналитикам) нужны все уровни требований, поскольку им приходится проверять систему на всех уровнях.

Использование wiki позволяло работать над требованиями параллельно всем членам проектной команды. Замечу, что в один и тот же момент разные

части требований находились в разных состояниях: от находящихся в работе до уже реализованных.

Бизнес-требования

Общие сценарии

Корневая страница нашего дерева требований состояла из общих сценариев, каждый из которых описывал один из 24 бизнес-процессов, подлежащих реализации в данном модуле. Сценарии на странице располагались в той последовательности, в которой они осуществлялись в компании: от создания объекта с проданными товарами, до передачи их клиенту. Некоторые специфические или вспомогательные сценарии помещались в конце в отдельном разделе.

Общий сценарий – это последовательность шагов пользователя и системы для достижения определенной цели. Описания общих сценариев были значительно менее формальны по сравнению со сценариями использования, поскольку они не предназначались для реализации. Основная цель общего сценария – это обобщить сценарии использования, подняться над системой и увидеть, что же в конечном итоге хочет сделать пользователь, и как система ему в этом помогает. Хочу заметить, что общие сценарии также содержали шаги, которые пользователь осуществлял вне системы, поскольку надо было отразить его работу во всей полноте, со всеми этапами, необходимыми для достижения бизнес-цели. На этом уровне хорошо видна роль системы в работе сотрудника компании, видно какая часть этой работы автоматизирована, а какая нет. Именно здесь становилось ясно, что некоторая последовательность действий, которую мы предлагали выполнить пользователю в системе, избыточна, что часть шагов можно сократить.

Некоторые другие цели общих сценариев:

- упорядочение знаний о работе пользователей и системы
- согласование бизнес-процессов с будущими пользователями

- основа для понимания того, что требования полны, что ничего не упущено
- входная точка при поиске нужного сценария или алгоритма

Вот пример одного из общих сценариев:

Печать документов, выполняет Специалист клиентского отдела

Цель данного этапа - печать и передача клиентам документов на груз

1. Пользователь производит пакетную печать документов на груз, т.е. выбирает **Задание на печать** из списка по номеру груза, переводит его в статус 'В работе' и печатает документы.
2. Пользователь проверяет корректность печати документов и переводит ЗНП в статус 'Готово'.
3. Если груз будет доставляться клиенту, СКО откладывает подготовленные документы в отдельную стопку, вечером эти документы раскладываются в ячейки для водителей (если будет осуществляться доставка по Москве) и в грузы (если груз поедет в регион).✅
4. Если клиент будет забирать груз самовывозом, СКО передает подготовленные документы представителю клиента.✅

Как видите, только половина шагов автоматизирована, да и те описаны как можно более кратко. Также из первого шага видно, что ручной перевод задания на печать в статус 'В работе' в принципе лишний, можно упростить работу пользователя и автоматически переводить задание в этот статус при печати.

Ссылка «Задание на печать», указывающая на описание объекта в системных требованиях, лишняя, поскольку никому не требуется перепрыгнуть на него из общего сценария. А вот ссылка «пакетная печать документов на груз» важна – она ведет на сценарий использования, формально описывающий действия пользователя и системы.

Наши сценарии использования имели следующий формат:

- Заголовок со следующими полями:
 - статус (В работе | Готов к рецензированию | Согласован)
 - пользователи (по описанию бизнес-ролей)
 - цель
 - предусловия
 - гарантированный исход
 - успешный исход
 - ссылка на описание пользовательского интерфейса (разработанного проектировщиком интерфейсов)
 - ссылка на сценарий тестирования (заполнялось тестировщиками)
- Основной сценарий
- Расширения сценария

Сценарии использования

Сценарий использования содержал пронумерованные шаги, которые в 99% случаев очевидным образом начинались со слов **Пользователь** или **Система**. Нумерация важна, поскольку позволяла в вопросах и комментариях сослаться на нужный пункт. Каждый шаг – это

обычно простое предложение в настоящем времени. Проверки и алгоритмы выносились на следующий уровень и часто на отдельные страницы, чтобы упростить восприятие сценария, а также для повторного использования.

Приведу сценарий использования, на который ссылается общий сценарий выше.

Пакетная печать документов на груз

Статус: **Согласовано**

Пользователи: Специалист клиентского отдела

Цель: Напечатать пакет документов на груз

Предусловия: Отображается список заданий на печать

Гарантированный исход: ЗнП остается в статусе 'Создано'

Успешный исход: Документы по грузам напечатаны, ЗнП переведено в статус 'Готово'.

Пользовательский интерфейс: Диалог "Пакетная печать по N грузам"

Чек-лист: Пакетная печать документов на груз (чеклист)

Основной сценарий

1. **Пользователь** выбирает задание на печать документов с статусе 'Создано' и нажимает кнопку 'В работу'.
2. **Система** в соответствии с [Алгоритм перехода ЗнП в статус 'В работе'](#) меняет статус ЗнП на 'В работе' и обновляет список.
3. **Пользователь** нажимает кнопку 'Пакетная печать'.
4. **Система** отображает [Диалог "Пакетная печать по N грузам"](#).
5. **Пользователь** выбирает Комплект.
6. **Система** отображает спецификацию комплекта.
7. **Пользователь** удостоверяется, что количество документов в комплекте задано верно; при необходимости, устанавливает нужное количество; нажимает кнопку 'Печать'.
8. **Система** запускает [Алгоритм формирования документов при пакетной печати](#) и печатает сформированный пакет документов на принтере.
9. **Пользователь** удостоверяется, что необходимые документы распечатаны и нажимает кнопку 'В Готовые'.
10. **Система** в соответствии с [Алгоритм перехода ЗнП в статус 'Готово'](#) меняет статус ЗнП на 'В Готовые' и обновляет список. ✓

Часто аналитики рисуют пользовательский интерфейс и на его основе пишут сценарии, объясняя это тем, что так нагляднее. Доля истины в этом есть, но мы придерживались позиции, что интерфейс – это дело проектировщика интерфейса. Сначала аналитик описывает, что должно происходить, а затем проектировщик интерфейса рисует эскиз web-страницы или диалога. При этом бывало так, что сценарий приходилось менять. В этом нет ничего страшного, ведь наша цель — спроектировать все части системы так, чтобы было удобно пользователю. При этом каждый участник проектной команды, будь то аналитик или проектировщик интерфейса, обладая специфическими знаниями и внося свой вклад в общее дело, оказывает влияние на работу других членов команды проекта. Только вместе, объединив усилия, можно получить отличный результат.

Алгоритмы и проверки

Интересная проблема возникла при написании алгоритмов. Аналитик пытался их описать как можно более полно, т.е. включать все возможные проверки и ответвления. Однако получившиеся тексты оказывались плохо читабельны, и, как правило, все равно какие-то детали упускались (вероятно, сказывалось отсутствие компилятора -). Поэтому аналитику стоит описывать алгоритм настолько полно, насколько это важно в плане бизнес-логики, второстепенные проверки программист сам обязан предусмотреть в коде. Например, рассмотрим простой алгоритм ниже.

Алгоритм перехода ЗНП в статус 'В работе'

Статус: **Согласовано**

Основной сценарий

1. Если **Статус** ЗНП = **Создано** или **Готово**, то **Система**:
 - a. Устанавливает **Пользователь** = <текущий пользователь>
 - b. Устанавливает **Статус** = **В работе**.
 - c. Пересчитывает **Количество грузоотправителей**.

В алгоритме указана всего одна проверка, но очевидно, что при написании кода метода программист должен реализовать проверки на входные

параметры; выбросить исключение, если текущий пользователь не определен и т.д. Также программист может объединить данный алгоритм с алгоритмами переходов в другие статусы и написать единый непубличный метод. На уровне API останутся те же операции, но вызывать они будут единый метод с параметрами. Выбрать лучшую реализацию алгоритмов – это как раз компетенция программиста.

Системные требования

Как известно, программирование – это разработка и реализация структур данных и алгоритмов. Таким образом, по большому счету, все, что надо знать программисту – это структуры данных, необходимые для реализации системы, и алгоритмы, которые ими манипулируют. При разработке системы мы использовали объектно-ориентированный подход, а поскольку в основе ООП лежат понятия класса и объекта, то наши структуры данных – это описания классов. Термин «класс» специфичен для программирования, поэтому мы использовали «объект». Т.о. объект в требованиях равен классу в объектно-ориентированном языке программирования (в скобках замечу, что в паре разделов требований пришлось изгаляться, чтобы в тексте разделить объект-класс и объект-экземпляр этого класса).

Описание каждого объекта располагалось на одной wiki-странице и состояло из следующих частей:

- Определение объекта (копия из списка терминов)
- Описание свойств объекта
- Описание операций и прав
- Данные
- Дополнительная информация

Все, что только можно, мы старались описать в табличном виде, поскольку таблица более наглядна, ее структура способствует упорядочению информации, таблица хорошо расширяема. Первая таблица каждого объекта

описывала признаки его свойств, необходимые для того, чтобы программист смог создать структуры данных в БД и реализовать объект на сервере приложения:

Название

Названием свойства оперирует как пользователь (например, «я изменил номер счета», Номер – свойство объекта Счет), так и проектная команда. Повсеместно в документации использовались ссылки на свойства в виде простой нотации Объект.Свойство, очевидной для любого участника проекта.

Тип

Были использованы Datetime, Date, Time, GUID, String, Enum, Int, Money, BLOB, Array(), Float, Timezone, TimeSpan. Тип имел отражение на всех уровнях приложения: на уровне БД, сервера приложения, в пользовательском интерфейсе в виде кода и графического представления. Каждому типу было дано определение, чтобы их реализация не вызывала вопросов у программистов. Например, было дано такое определение типу Money: содержит вещественное число с точностью до 4-го знака после запятой, число может быть отрицательным и положительным; одновременно со значением система хранит валюту; валюта по умолчанию — российский рубль.

Признак редактируемости

Да или **Нет** в зависимости от того, позволяет ли система пользователям менять значение этого свойства в операции редактирования. В описываемой системе это ограничение реализовывалось на сервере приложения и в пользовательском интерфейсе.

Признак наличия нуля

Да или **Нет** в зависимости от того, может ли поле не содержать значения. Например, поле типа **Bool** должно содержать одно из возможных значений, а поле типа String обычно может быть пустым (**NULL**). Это ограничение реализовывалось на уровне БД и на сервере приложения.

Признак уникальности

Да или **Нет** в зависимости от того, является ли это поле уникальным. Часто уникальность определяется на группе полей, в этом случае у всех полей в группе стояло **Да+**. Это ограничение реализовывалось на уровне БД (индекс) и на сервере приложения.

Комментарий

Описание поля: что означает, для чего нужно, как используется. Если значение свойства вычисляемое, то это указывается явно с описанием алгоритма расчета этого значения.

Кроме этих было еще две колонки, которые заполнялись программистами серверной части при реализации объекта:

- Название свойства объекта в программном интерфейсе.
- Название поля в БД.

Оба этих поля не обязательные, поскольку, например, свойство объекта может не храниться в БД, а быть вычисляемым, как сумма счета.

Еще раз обратим внимание, что в написании требований принимали участие программисты. Это важно по многим причинам. Во-первых, таким образом программисты лучше осознавали требования, более того, требования становились «ближе к телу», а не просто неким куском бумаги, написанным каким-то аналитиком. Во-вторых, автоматически формировалась документация для API. В-третьих, поддерживалась трассируемость (traceability) требований, т.е. всегда было понятно, реализовано ли то или иное свойство, что особенно становилось важным при модификации требований. Безусловно, такая методология требовала большей дисциплины от программистов, что на самом деле являлось положительным фактором.

Кроме того благодаря этим колонкам, программистам, работающим над разными уровнями приложения, всегда можно было найти общий язык, т.е. понять соответствие между свойством объекта в требованиях, полем в базе данных и свойство в API.

Как уже было сказано, табличный вид очень удобен для расширения. Например, для описания начальной миграции, в рассматриваемом примере, была колонка с именем свойства старой системы или алгоритмом преобразования данных. Также мы использовали специальные значки для описания того, как выглядит объект в пользовательском интерфейсе. Одно время у нас была колонка для имени индекса в БД, чтобы программисты не забывали их создавать для уникальных полей. При необходимости вы можете добавить колонку с размерностью типов данных для каждого свойства.











Вот типичное описание свойств объекта.

Задание на печать документов

Статус: Согласовано

Задание на печать документов - объект системы, предназначенный для отслеживания готовности документов и содержит информацию о грузе и грузоотправителях.

Свойства задания на печать:

UI	PrintTaskDto	Свойство	Тип	Ред?	Ноль?	Уник?	Комментарий	БД: PrintTask
	ID	Идентификатор	GUID	Нет	Нет	Да	Уникальный идентификатор задания.	ID
	Number	Номер	Int	Нет	Нет	Да	Уникальный номер задания для его идентификации пользователем.	Number
	AdmUser	Пользователь	GUID*	Нет	Да		Пользователь, который взял задание в работу.	AdmUserID
	PrintTaskStatus	Статус	Enum	Нет	Нет		Статус задания на печать. Изменение статуса происходит не операцией редактирования, а специальными операциями перевода задания по статусам.	PrintTaskStatus_EN
	Cargo	Груз	GUID*	Нет	Нет		Груз, по которому необходимо подготовить документы на печать.	CargoID
	VCustomer	Клиент	GUID*				Вычисляется и равно Груз.Клиент.	
	ShipperCount	Количество грузоотправителей	Integer				Вычисляется при создании задания и при переводе его по статусам. Равно количеству юрлиц Продавец счетов, неудаленные позиции которых присутствуют в грузе.	
 		Способ отправки документов	Enum				Вычисляется и равно Груз.Способ отправки документов	
  		Грузоотправители	Array(GUID*)				Вычисляется и равно Груз.Грузоотправители	
  		Накладные	Array(GUID*)	Нет	Да		Вычисляется и равно Груз.Накладные	

Вторая таблица объекта содержала описание его операций и их прав. Каждая операция в системе имела уникальное название (колонка **Операция**), но в пользовательском интерфейсе (в меню) операции отображались под краткими названиями (**Краткое название**). Для выполнения любой операции надо было обладать определенным правом (**Право**). Колонка **Комментарий** для сложных методов содержала описание алгоритма или ссылку на него или на более общий сценарий использования. CRUD операции над всеми типами объектами, в данном примере, были стандартизированы, поэтому для них алгоритмы обычно не требовались.

Колонка **Название в коде** опять заполнялась программистом что, как и при описании объекта, было нужно для документирования API, повышения вовлеченности программистов в написание требований и трассируемости. Далее – пример описания операций объекта:

Операции и права

Система позволяет выполнять следующие операции над заданиями на печать:

Операция	Краткое название	Право	Комментарий	Название в коде
Просмотр задания на печать	Открыть	Просмотр заданий на печать		PrintTaskGetRequest
Получение списка заданий на печать	Получить список	Просмотр заданий на печать		PrintTaskGetListRequest
Перевод задания на печать в статус 'В работе'	В работу	Перевод заданий на печать в статус 'В работе'	Алгоритм перехода ЗНП в статус 'В работе'	PrintTaskSetInProgressRequest
Перевод задания на печать в статус 'Готово'	В Готовые	Перевод заданий на печать в статус 'Готово'	Алгоритм перехода ЗНП в статус 'Готово'	PrintTaskSetReadyRequest
Перевод задания на печать в статус 'Создано'	Отменить	Перевод заданий на печать в статус 'Создано'	Алгоритм перехода ЗНП в статус 'Создано'	PrintTaskSetCreatedRequest
Получение списка грузоотправителей	Получить список	Просмотр юридических лиц		PrintTaskShipperGetListRequest
Получение списка накладных на отпуск по заданию на печать	Получить список	Просмотр накладных		PrintTaskInvoiceGetListRequest
Пакетная печать документов по грузам	Пакетная печать	Пакетная печать документов	Алгоритм формирования документов при пакетной печати	PrintTaskPrintBatchRequest

В этом разделе были также таблицы, описывающие переход по статусам.

Система поддерживает следующие статусы задания:

Значение	Описание	Порядок сортировки	Активно?	XEnums.PrintTaskStatus
Создано	Начальный статус, задание не обработано.	1	1	Created
В работе	Задание находится в работе.	2	1	InProgress
Готово	Задание выполнено.	3	1	Ready

Система не позволяет добавлять, изменять или удалять статусы задания.

Ниже приведена диаграмма возможных переходов задания между статусами и соответствующие операции:

Исходный статус \ Целевой статус	Создано	В работе	Готово
Создано	-	В работу	-
В работе	Отменить	-	В Готовые
Готово	-	В работу	-

В ячейках стоит краткое название операции, которая переводит исходный статус в целевой. Для более сложных случаев приходилось рисовать полноценные диаграммы.

После инсталляции системы в ней должны присутствовать определенные данные. Например, пользователь с администраторскими правами или список стран согласно общероссийскому классификатору стран мира. Эти данные описывались в разделе **Данные**.

Все, что не уложилось в стандартные разделы выше, шло в раздел **Дополнительная информация**. Например, у нас это была информация по связям данного объекта со старой системой.

Подытоживая, можно сказать, что системные требования для объекта содержали всю необходимую информацию для его реализации программистом: структуру данных в БД, описание доменного объекта, ограничения на данные и методы, алгоритмы реализации методов, данные, которые должны быть при инсталляции системы. Структура описания проста для понимания и расширяема.

Возможно кому-то может показаться, что такая детализация требований отнимает много времени и не нужна. Однако без этого, программист должен будет догадываться, что конкретно необходимо реализовать? Разве фразы «Необходимо реализовать объект Пользователь» достаточно, чтобы через некоторое время получить работающий код? Сколько надо выпить чая с аналитиком, чтобы вытащить из него данные по 40 (столько было у нас) свойствам пользователя? Кто, как не аналитик или проектировщик, должен приложить усилия и описать все объекты?

[Постановка задач программистам](#)

После описания того, как выглядят требования, рассмотрим интересный вопрос: как должны быть сформулированы задачи для программистов (ограничимся серверной частью многозвенного приложения)? Оказывается, большинство задач (не дефектов) сводится к трем вариантам:

Типовая задача 1

Заголовок: Реализовать такой-то объект. Текст задачи — ссылка на страницу с системными требованиями к объекту.

В такой задаче программисту необходимо:

- создать структуры в БД (таблица, ключи, индексы, триггеры и т.д.);
- реализовать доменный объект;
- реализовать создание начальных данных.

Все это возможно сделать на основе описания объекта в системной части требований. Программист также должен дополнить таблицу с описанием свойств названиями полей таблицы БД и объекта API.

Типовая задача 2

Заголовок: Реализовать такую-то операцию такого-то объекта и права на нее. Текст задачи — ссылка на страницу с системными требованиями к объекту. Программист находит на странице название операции и права, а по ссылке в колонке Комментарий — алгоритмы, проверки, сценарий использования.

Типовая задача 3

Заголовок: Скорректировать объект и/или операцию. Данная задача необходима в случае изменений требований. Текст задачи содержит описание изменений или ссылку на страницу сравнения версий требований.

[Инструмент для написания и управления требованиями](#)

Как, возможно, многие догадались, для работы с требованиями мы использовали Atlassian Confluence. Хочу кратко перечислить достоинства этого продукта.

- Удаленная работа. Собственно, как и у любой wiki.
- Ссылки. Как вы видели выше, ссылки для нас — один из основных инструментов для связывания отдельных частей требований.

- Возможность дробить требования на части (каждая часть – на своей странице).
- Оповещения при изменении. Это одно из важнейших средств совместной работы. Например, получив такое оповещение по одному из сценариев, руководитель разработки может ставить задачи разработчикам, а тестировщики знает, что надо скорректировать сценарии тестирования.
- Комментарии. Многие страницы требований у нас обрастали развесистыми иерархиями комментариев. В Confluence работать с ними достаточно удобно, поскольку иерархия не плоская, а в виде дерева. Кроме того есть возможность использовать полноценный редактор, а не просто текст.
- Наличие мощного текстового редактора. Не буду здесь подробно останавливаться, отмечу лишь, что на всем протяжении нашей работы Atlassian совершенствовал редактор, и если вначале было достаточно много глюков, то затем подавляющее большинство из них было исправлено.
- Хранение истории, сравнение разных версий страниц, возможность отката на старую версию.
- Просмотр иерархии страниц в виде дерева.

Однако было и несколько проблем:

- Поскольку все требования используют одни и те же названия объектов и их свойств, то было бы очень удобно иметь инструмент, который при изменении названия менял его во всей документации. А при удалении – находил все, уже недействительные, ссылки на него.
- Не было возможности сбора статистики. Например, каждое требование имело статус, но мы не могли автоматически собирать статусы всех требований и иметь динамическую картину процесса разработки требований. Но, кажется, на данный момент что-то подобное в Confluence уже появилось.
- Диаграммы приходилось рисовать в другой системе, сохранять в PNG и уже картинку помещать на страницу Confluence. При этом еще надо было приложить исходник, чтобы через пару месяцев его можно было найти и поправить.
- Я не нашел способа экспортировать иерархию страниц в MS Word. Экспорт в XML и PDF очень часто глючил (возможно, дело в размере иерархии).

Антон Стасевич [Пример написания функциональных требований к Enterprise-системе <https://habr.com/ru/post/245625/>]

Задание

Создать краткое описание объекта автоматизации, сформулировать основные задачи автоматизации объекта, описать основные параметры проектируемой информационной системы, описание путей достижения целей.

Вторым этапом, сформулировать требования к информационной системе аналогично приведенному примеру.

Создать отчет, оформить его в соответствии с требованиями ГОСТ 7.32-2017

Работа 2 выбор (эскизное проектирование) архитектуры системы

Цель эскизного проектирования — сформировать варианты автоматизированной системы, сравнить их и предложить заказчику наилучший, по мнению проектировщика, вариант.

Централизованная (я) и децентрализованная (б) компоновки гибкого

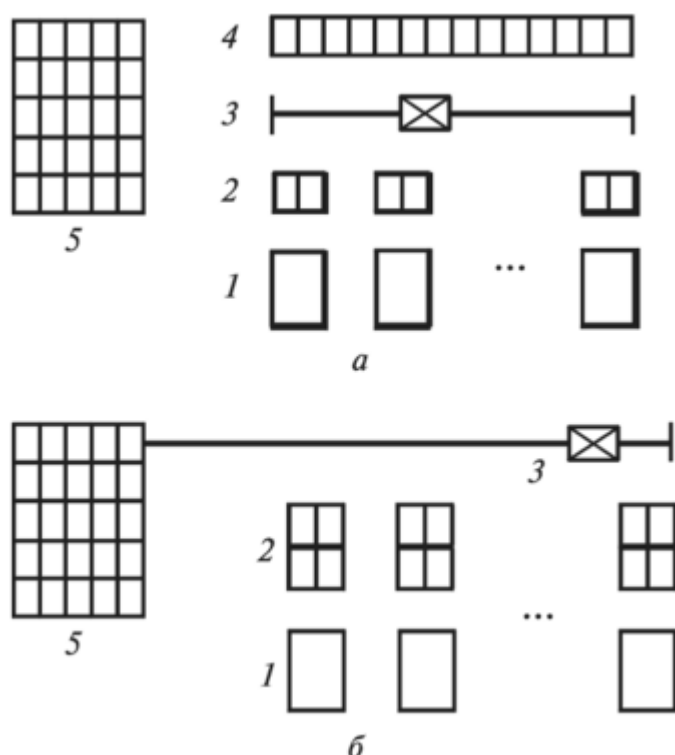
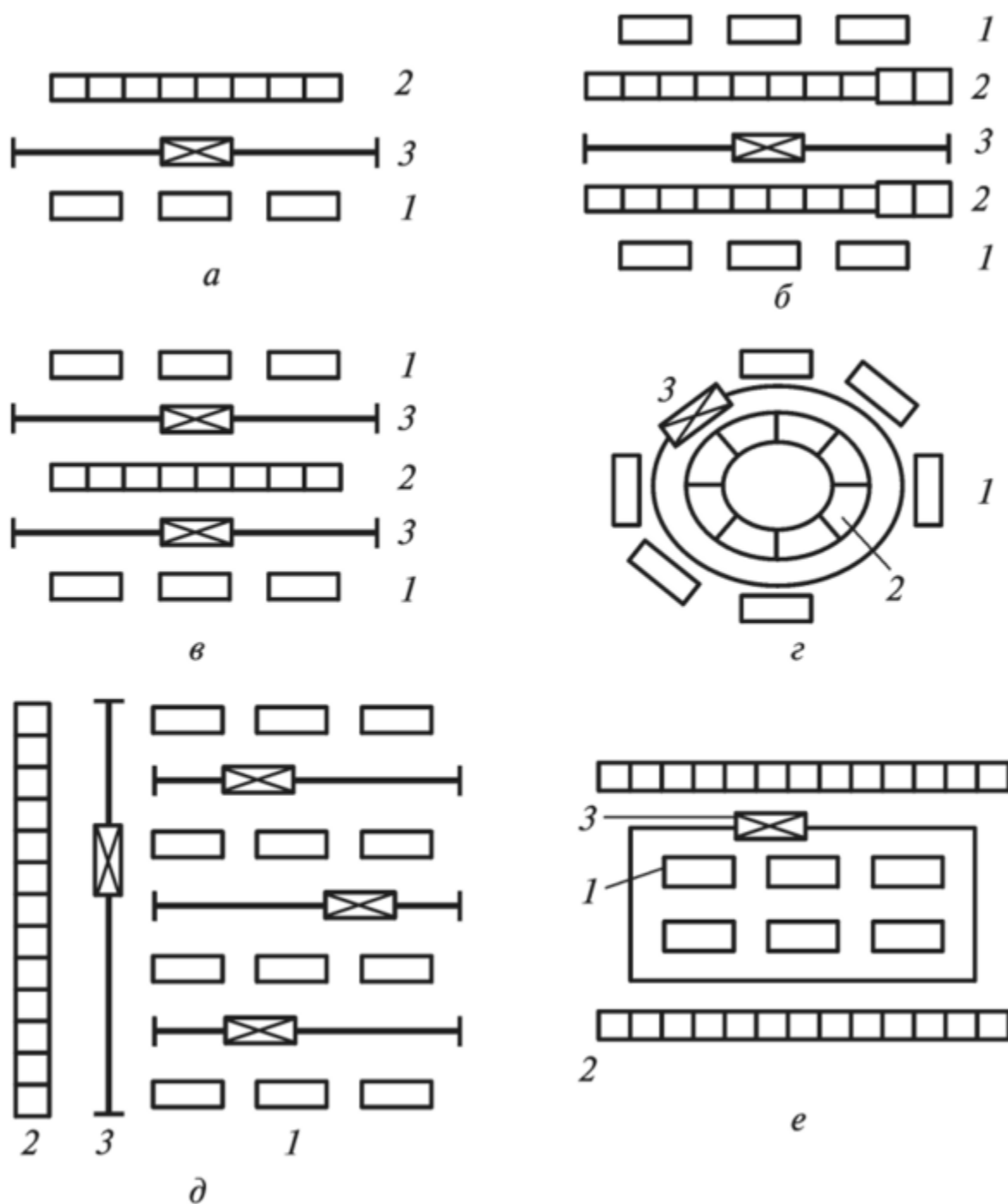


Рис. 12.3. Централизованная (я) и децентрализованная (б) компоновки гибкого производства

Гибкое производство состоит из технологической, транспортной, складской, управляющей и контрольной подсистем, связанных между собой материальными и информационными потоками. При централизованной компоновке гибкого производства (рис.12.3, а) применяют местный склад 4, а у каждой единицы оборудования 1 размещают накопитель 2 небольшой емкости. Доставку заготовок и полуфабрикатов осуществляет автоматически управляемая тележка 3, курсирующая между местным складом и

накопителями. Между местным 4 и общим 5 складами независимо от технологического процесса ведется обмен заготовками и готовыми изделиями.



Р и с. 12.4. Примеры структур гибкого производства:

а — с однорядным расположением единиц оборудования и внешним складом; б — с двухрядным расположением единиц оборудования и внутренним складом; в — с параллельной работой транспортных роботов; г — с круговым обслуживанием; д — с последовательной работой транспортных

роботов; е — с круговым обслуживанием и многорядным расположением складов лями. При децентрализованной компоновке (см. рис. 12.3, б) местный склад отсутствует, а возле каждой единицы оборудования размещают накопители 2 заготовок и полуфабрикатов, емкость которых должна быть достаточна для автономной работы единицы оборудования в интервале между рейсами тележки к общему складу и обратно.

Комбинации вариантов расположения единиц оборудования и способов их обслуживания транспортными тележками приводят к разнообразию структур гибкого производства. Некоторые из них показаны на рис. 12.4.

Распределенные в пространстве единицы технологического оборудования в гибком производстве связаны между собой транспортными средствами, образующими автоматизированную транспортно-складскую систему. Управление системой должно обеспечивать оперативное планирование и регулирование материальных потоков, оптимизацию маршрутов перемещения и мест хранения грузов, взаимодействие единиц технологического и транспортно-складского оборудования. На нижнем уровне осуществляются ввод и вывод информации о состоянии и кодах грузов и единиц оборудования, контроль перемещений грузов, остановка транспортных устройств в заданных местах, логическое управление исполнительными устройствами. При вводе задания крану-штабелеру указывают стеллаж, ярус и ячейку в ярусе. На рис. 12.5 показаны варианты транспортных связей между единицами оборудования и единицами транспортно-складской системы. В схеме (а) кран-штабелер 2 отыскивает грузы в ячейках центрального 1 и промежуточного 3 складов, затем передает их транспортному роботу 5, который доставляет грузы к единицам оборудования 4.

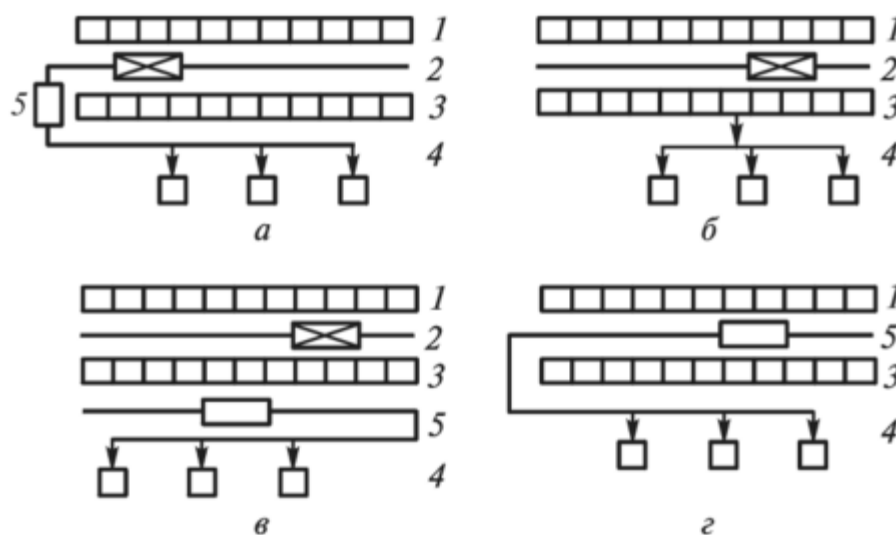


Рис. 12.5. Варианты транспортно-складских систем гибкого производства

схемы (б) грузы с центрального склада 1 передаются штабелером 2 в промежуточный склад 3 гравитационного типа, из которого они поступают сразу к накопителям у единиц оборудования 4. Схема (в) отличается от схемы (б) доставкой грузов от склада 3 транспортным роботом 5. В схеме (г) транспортный робот 5 доставляет грузы от центрального 1 и промежуточного 3 складов к единицам оборудования. Вариант автоматизированной транспортно-складской системы выбирают по минимальному среднему пути транспортного средства при выполнении одноадресных заявок:

$$S = \frac{\left(\sum_{i=1}^L S_i \right)}{L} \rightarrow \min,$$

где S_j — пробег при выполнении j -й заявки; $L = \tau_2$ — число размещений из τ специализированных единиц оборудования с повторениями.

Для сокращения простоев технологической линии в случаях остановок единиц оборудования между предыдущей А и последующей В единицами размещают промежуточный накопитель. Запас деталей Z в накопителе должен быть таким, чтобы суммарные затраты на хранение и потери от простоя станка

из-за прекращения предыдущей операции были минимальны. Время простоя станка А при устранении отказа станка В составит:

$$t = \begin{cases} 0, & \text{если } Z \geq PT; \\ \frac{PT - Z}{P}, & \text{если } Z < PT, \end{cases}$$

где Т — время устранения отказа станка В Р— частота подачи полуфабрикатов на станок В.

При выборе программно-аппаратных средств автоматизации проектировщик должен исходить из технологических требований (табл. 12.1).

Сначала определяют задачу, для решения которой будет применено средство автоматизации, например, включение режима торможения транспортного робота. Затем задают технологические требования к средству, например, диапазон и погрешность измерения уровня жидкости. Определяют место размещения этого средства. В остальных столбцах указывают тип, изготовителя и стоимость средства.

Задача	Требования	Место установки	Тип	Изготовитель	Стоимость
Определение положения транспортного робота у станка	Точность 5 мм; расстояние срабатывания > 100 мм; выход — бинарный; реагирование на металл	На станке			

Одна и та же задача может быть решена десятками имеющихся на рынке средств автоматизации разных фирм. Это могут быть датчики, программируемые контроллеры, исполнительные устройства, выпускаемые десятками мировых производителей и имеющие близкие характеристики. Рассмотрим задачу выбора проектировщиком технических средств для компоновки автоматизированной системы. Каждое из технических средств

имеет некоторую стоимость C_j и степень полезности для проектировщика. При оценке полезности средства автоматизации следует учитывать степень выполнения требований технического задания, соответствие технологическим условиям и обеспечение совместимости средств разных фирм. Если в техническом задании требуется разработать систему с минимальными энергозатратами, то степенью полезности для проектировщика будет потребление энергии рассматриваемым средством. Например, для объекта проектирования с питанием от бортовых источников степенью полезности будет величина потребления энергии. Суммарные затраты на приобретение набора средств не должны превышать выделенную сумму N . Проектировщик должен выбрать набор средств, обладающий максимальной полезностью. Для выбора может быть применен метод ветвей и границ.

Процедуру поиска покажем на примере выбора оптимального набора из пяти средств, имеющихся на рынке. Упорядочим предлагаемые средства в порядке убывания отношения a_j/c_j (табл. 12.2).

Таблица 12.2. Упорядоченный набор средств автоматизации

x_j	1	2	3	4	5
a_j	6	8	10	11	9
c_j	2	3	7	10	9

На приобретение набора выделено 20 условных единиц, поэтому введем ограничение по стоимости:

$$N \leq 20.$$

Если приобрести первое средство (x_1), то в пределах выделенной суммы можно приобрести и другие средства, кроме 4-го и 5-го:

$$Q_1 = 2 \cdot 1 + 3 \cdot 1 + 7 \cdot 1 + 10 \cdot 0 + 9 \cdot 0 \leq 20.$$

Для сочетания $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 0, x_5 = 0$ рассчитаем значение полезности:

$$f_1 = 6 \cdot 1 + 8 \cdot 1 + 10 \cdot 1 + 11 \cdot 0 + 9 \cdot 0 = 24.$$

Если не приобретать 1-е средство ($x_1 = 0$), то можно приобрести 2-е, 3-е и 4-е средства, поскольку:

$$Q_2 = 2 \cdot 0 + 3 \cdot 1 + 7 \cdot 1 + 10 \cdot 1 + 9 \cdot 0 \leq 20.$$

Тогда:

$$f_2 = 6 \cdot 0 + 8 \cdot 1 + 10 \cdot 1 + 11 \cdot 1 + 9 \cdot 0 = 29.$$

Поскольку $f_2 > f_1$, выбираем вариант отказа от первого средства. Для этого варианта рассчитываем стоимость и полезность приобретения 2-го средства в случае приобретения всех остальных средств:

$$Q_3 = 2 \cdot 0 + 3 \cdot 1 + 7 \cdot 1 + 10 \cdot 1 + 9 \cdot 0 \leq 20;$$

$$f_3 = 6 \cdot 0 + 8 \cdot 1 + 10 \cdot 1 + 11 \cdot 1 + 9 \cdot 0 = 29.$$

Если отказаться от покупки второго средства, то:

$$Q_4 = 2 \cdot 0 + 3 \cdot 0 + 7 \cdot 1 + 10 \cdot 1 + 9 \cdot 0 \leq 20;$$

$$f_4 = 6 \cdot 0 + 8 \cdot 0 + 10 \cdot 1 + 11 \cdot 1 + 9 \cdot 0 = 21.$$

Поскольку $29 > 21$, второе средство целесообразно приобрести. С учетом уже принятых решений выбираем решение для 3-го средства: для приобретения ($x_3 = 1$)

$$f_5 = 6 \cdot 0 + 8 \cdot 1 + 10 \cdot 1 + 11 \cdot 1 + 9 \cdot 0 = 29,$$

для отказа от покупки ($x_3 = 0$)



Рис. 12.6. Схема выбора средств автоматизации методом ветвей и границ

$$f_6 = 6 \cdot 0 + 8 \cdot 1 + 10 \cdot 0 + 11 \cdot 1 + 9 \cdot 0 = 19.$$

Если приобрести x_4 , то $f_7 = 29$. Если не приобретать x_4 , то $f_8 = 27$. Приобретение x_5 нецелесообразно из-за превышения выделенных средств ($29 > 20$).

Таким образом, максимальную полезность обеспечит выбор средств 2-го, 3-го и 4-го (рис. 12.6).

Предварительное ранжирование средств позволяет пожертвовать в первую очередь тем средством, для которого отношение a_j/c_j минимально. После принятия решения относительно некоторого средства решение должно сохраняться при переходе к рассмотрению последующих средств. Применение метода ветвей и границ позволяет выбрать правильное решение относительно целесообразности приобретения некоторого средства с учетом наилучших решений относительно целесообразности приобретения последующих средств. [https://studref.com/387970/tehnika/razrabotka_eskiznogo_proekta#631]

Техническое задание.

Результатом выполнения данной стадии является техническое задание, оформленное в соответствии с ГОСТ 19.105—78 (изм. 09.1981.) «Общие требования к программным документам» и ГОСТ 19.106—78 «Общие требования к программным документам, выполненным печатным способом на листах формата 11 и 12» (по ГОСТ 2.301—68).

Эскизный проект.

Конкретное содержание работ стадии эскизного проекта и их объем определяют степень сложности разрабатываемого ПО. Результатом выполнения данной стадии является полное описание архитектуры ПО. Как правило, это описание делается на нескольких уровнях иерархии. На верхнем уровне детализации выделяются основные подсистемы, которым присваиваются имена, устанавливаются связи между подсистемами, их функции, получаемые путем декомпозиции предполагаемых функций ПО. Затем процедура декомпозиции выполняется для каждой подсистемы, выделяются модули, составляющие данную подсистему. В конечном итоге получается иерархически организованная система, состоящая из уровней, каждый из которых представляет собой совокупность взаимосвязанных модулей.

Единицы, выделяемые на различных иерархических уровнях функциональной архитектуры системы, определяются по усмотрению разработчика. Стандарты ЕСПД различают программные единицы только с точки зрения их документирования.

Результаты эскизного проекта отображаются в документе «Пояснительная записка к эскизному проекту», оформленному в соответствии с ГОСТ 19.105—78 и ГОСТ 19.404—79.

После утверждения пояснительной записки она становится программным документом, правила дублирования, учета, хранения которого

определяется ГОСТ 19.601—78 «Общие правила дублирования, обращения, учета и хранения» и ГОСТ 19.602—78 «Правила дублирования, учета и хранения программных документов, выполненных печатным способом». Последующие стадии и этапы разработки ПО могут выявить необходимость внесения изменений в ЭП. Эти изменения должны быть отражены в пояснительной записке в соответствии с ГОСТ 19.603—78 «Общие правила внесения изменений в программные документы» и ГОСТ 19.602—78 «Правила внесения изменений в программные документы, выполненные печатным способом».

В качестве примера приведем фрагмент расширенного перечня работ стадии эскизного проекта [15]:

- разработка плана совместных работ на разработку ПО;
- разработка и обоснование математической модели системы на ЭВМ и описание результатов моделирования;
- разработка и обоснование алгоритмов и временных графиков функционирования ПО по всем режимам работы;
- разработка и обоснование ресурсов памяти для реализации алгоритмов;
- разработка перечня документов на ПО;
- разработка и обоснование структуры БД, внешних входных и выходных данных;
- разработка и обоснование алгоритмов информационного обеспечения;
- определение взаимосвязей между видами программ;
- разработка и обоснование набора тестов для проверки ПО;
- разработка и обоснование организации наращивания и развития ПО;
- оформление пояснительной записки и ведомости эскизного проекта ПО (в соответствии с ГОСТ 19.105—78, ГОСТ 19.404—79 и ГОСТ 2.106—68 ЕСКД. «Текстовые документы»);

• согласование и утверждение ЭП.
[https://studref.com/528098/informatika/tehnicheskoe_zadanie#315]

РД 50-34.698-90 Пояснительная записка к эскизному проекту на создание автоматизированной системы (пример эскизного проекта)

Ниже представлен пример (образец) проектного документа "Пояснительная записка к эскизному проекту на создание автоматизированной системы", основанный на методических указаниях РД 50-34.698-90.

Данный документ формируется ИТ-специалистом на стадии эскизного проектирования информационной системы.

В качестве примера разработки информационной системы взят проект внедрения информационно-аналитической системы «Корпоративное хранилище данных».

На странице ниже приведен содержание пояснительной записки эскизного проекта в соответствии с ГОСТ, внутри каждого из разделов кратко приведены требования к содержанию и текст примера заполнения (выделен вертикальной чертой).

Разделы пояснительной записки:

Общие положения

Основные технические решения

Решения по структуре системы, подсистем, средствам и способам связи для информационного обмена между компонентами системы

Решения по взаимосвязям АС со смежными системами, обеспечению ее совместимости

Решения по режимам функционирования, диагностированию работы системы

Решения по персоналу и режимам его работы

Сведения об обеспечении заданных в техническом задании потребительских характеристик системы, определяющих ее качество

Состав функций, комплексов задач реализуемых системой

Состав и размещение комплексов технических средств

Решения по составу информации, объему, способам ее организации, видам машинных носителей, входным и выходным документам и сообщениям, последовательности обработки информации и другим компонентам

Методы и средства разработки

Мероприятия по подготовке объекта автоматизации к вводу системы в действие

Пояснительная записка к эскизному проекту на создание автоматизированной системы «Корпоративное хранилище данных»

1. Общие положения

1.1. Наименование системы

1.1.1. Полное наименование системы

Полное наименование - корпоративное хранилище данных.

1.1.2. Краткое наименование системы

Краткое наименование - КХД, Система.

1.2. Основания для проведения работ

Указывается номер и дата договора.

Перечень документов, на основании которых создается система, кем и когда утверждены документы.

Например:

Работа выполняется на основании договора № ... от ..., заключенного между ...

1.3. Наименование организаций – Заказчика и Разработчика

1.3.1. Заказчик

Заказчик: ОАО Заказчик

Адрес фактический: г. Москва ...

Телефон / Факс: +7 (495) 2222222

1.3.2. Разработчик

Разработчик: ЗАО Разработчик

Адрес фактический: г. Москва ...

Телефон / Факс: +7 (495) 3333333

1.4 Цели, назначение и область использования системы

Определяются цели (чего хочет достичь организация Заказчика от внедрения системы); назначение (для каких пользователей предназначена); области использования АИС (какие виды деятельности организации Заказчика охватывает система).

Информация для разделов "Наименование системы", "Основания для проведения работ", "Наименование организаций Заказчика и Разработчика", "Цели, назначение и область использования системы" берется из одноименных разделов технического задания на создание корпоративного хранилища данных.

1.5. Нормативные ссылки

При эскизном проектировании использовались следующие нормативно-технические документы:

Например:

1. Техническое задание на создание информационной системы КХД

2. ГОСТ 34 -...

3. ...

1.6 Очередность создания системы

Указывается очередность создания системы и характеристики каждой очереди (функциональность, ограничения, сроки, исполнители).

Например:

Ниже представлена предполагаемая очередность создания системы:

- Производится разработка модели хранилища данных.
- Согласовываются форматы и структуры обмена данными с системами-источниками.
- Проектируются процессы сбора данных в область временного хранения данных.
- Проектируются процессы загрузки данных в область постоянного хранения данных.
- Проектируются типовые отчеты.
- Разрабатывается схема организации доступа пользователей.
- Производится настройка активного сетевого оборудования.
- Производится настройка аппаратно-технической части: установка и настройка серверов, подключение к сетевому активному оборудованию, настройка сетевых параметров и т.п.
- Разрабатывается план установки серверного программного обеспечения.
- Производится установка серверного программного обеспечения.
-
- Реализация ...
- ...

- Тестирование ...

2. Основные технические решения

2.1. Решения по структуре системы, подсистем, средствам и способам связи для информационного обмена между компонентами системы

2.1.1 Логическая и компонентная архитектура системы

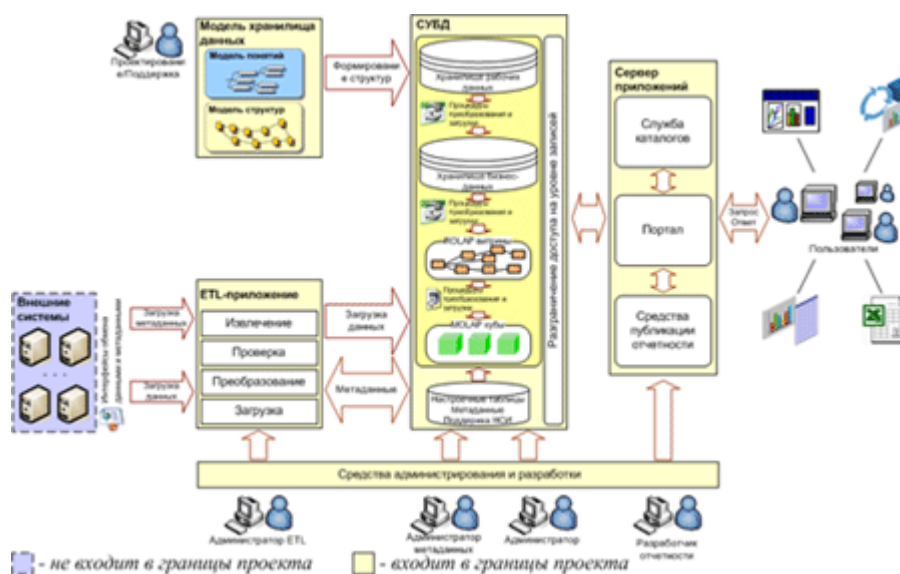
Приводится перечень, назначение и взаимосвязи готовых (закупаемых) и вновь разрабатываемых программных компонентов системы.

Например:

Перечень используемых для создания системы КХД программных средств приведен ниже:

- СУБД (название, версия);
- ETL приложение (название, версия);
- BI приложение (название, версия).

Логическая и компонентная архитектура системы представлена на рисунке ниже.



Пояснительная записка к эскизному проекту на создание автоматизированной системы - Архитектура системы

В состав разрабатываемой системы будут включены следующие технологические компоненты:

- программное обеспечение поддержки модели данных;
- ETL-приложение – это комплексное решение, с помощью которого реализуются процессы извлечения, проверки, преобразования и загрузки данных.
- сервер БД представляет собой промышленную систему управления базами данных.
- сервер приложений – продукт, обеспечивающий поддержку промышленной инфраструктуры бизнес-приложений. Включает в себя следующий ряд приложений, обеспечивающих стандартные подходы к организации служб каталогов; развертывание сервисов анализа и отчетности.
- средства администрирования и разработки – набор программных продуктов, предназначенных для администрирования системы ETL, базы данных, сервера приложений и разработки отчетности и дополнительных приложений.
- клиентские места сотрудников (внутри локальной вычислительной сети), представляющие собой автоматизированные рабочие места.

2.1.2. Функциональная структура системы

В данном разделе формируется схема функциональной структуры системы КХД. Схема формируется следующим образом: в виде общего прямоугольника изображается система КХД, далее в этот прямоугольник вставляются прямоугольники, обозначающие подсистемы. Внутри каждой подсистемы формируется перечень функций, которые она выполняет (перечень подсистем и перечень выполняемых ими функций берутся из раздела технического задания «Требования к функциям, выполняемым системой»). После этого на основании требований, изложенных в пункте технического задания «Требования к информационному обмену между

компонентами системы», прорисовываются связи между подсистемами и связи подсистем с внешними информационными системами и пользователями (к каким подсистемам обращаются пользователи). Возле каждой подсистемы схематично изображается её администратор.



Например:

Пояснительная записка к эскизному проекту на создание автоматизированной системы - Схема функциональной структуры

Ниже рисунка приводится описание каждой подсистемы. Описание берется из пункта «Требования к структуре и функционированию системы» технического задания. Описание подсистем может быть скорректировано.

Затем производится описание взаимосвязей между подсистемами. Описание взаимосвязей формируется путем ответа на вопрос: «Какой процесс определяет взаимосвязь между каждой из подсистем?».

Например:

Связь «Подсистема сбора, обработки и загрузки данных - Подсистема хранения данных» определяет процесс загрузки данных в ХД. Загрузка данных происходит по протоколу <указать протокол> в определенные временные интервалы и с заданной периодичностью.

После описания взаимосвязей подсистем в табличной форме приводится описание связей «Подсистема-Пользователь». В данной таблице отражается

информация о том, какой администратор/пользователь работает с какой подсистемой - в матрице ставится крестик на нужном пересечении Подсистема-Пользователь.

2.2. Решения по взаимосвязям АС со смежными системами, обеспечению ее совместимости

Определяются решения по взаимосвязям системы КХД со смежными системами, обеспечению ее совместимости (описание используемых протоколов обмена данными и средства и методы обмена данными). За основу берутся данные из пункта «Требования к характеристикам взаимосвязей со смежными системами» технического задания.

Пример:

Приводится перечень смежных систем, способы их взаимодействия.

Наименование смежной системы	Предпочтительный способ взаимодействия
Информационная система управления предприятием	Использование ПБД
Информационно-справочная система	Файлы ОС определенного формата
...	...

Ниже представлена общая схема взаимодействия системы КХД и смежных систем.

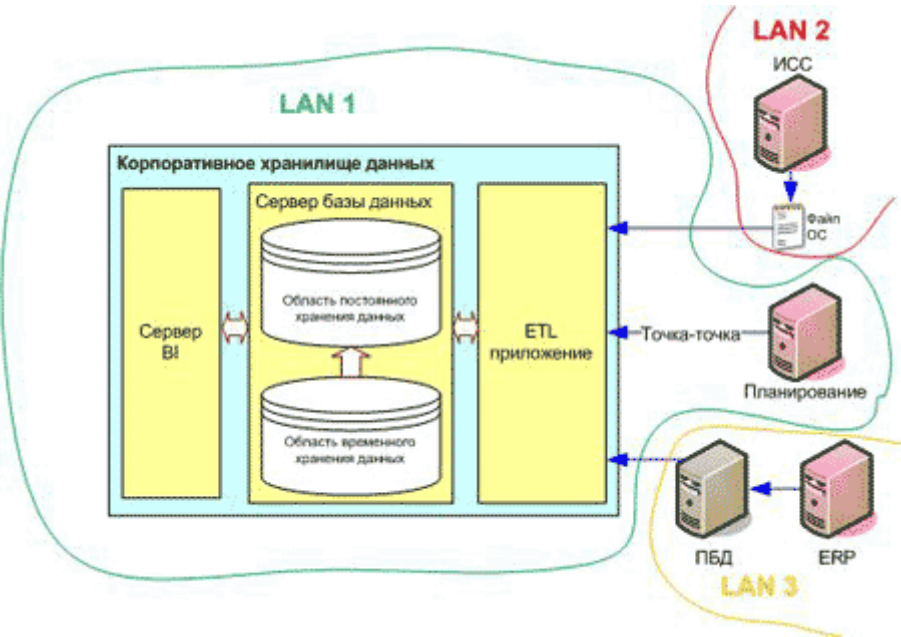


Схема взаимодействия. Пояснительная записка к эскизному проекту

2.3 Решения по режимам функционирования, диагностированию работы системы

На основании пункта «Требования к режимам функционирования» технического задания приводятся режим работы системы КХД.

Также приводится описание решений по диагностированию системы, осуществляемых путем установления и изучения признаков, которые характеризуют состояние системы, для предсказания возможных отклонений и предотвращения нарушений нормального режима ее работы.

Например:

Предлагается следующая реализация решений по режимам функционирования системы:

- Основной режим, в котором все подсистемы выполняют свои основные функции.

- Профилактический режим, в котором одна или все подсистемы не выполняют своих функций. В данный режим работы система переходит в следующих случаях: возникновение необходимости модернизации аппаратно-программного комплекса; возникновение необходимости проведения технического обслуживания; выход из строя аппаратно-программного комплекса, вызванный выходом из строя элементов аппаратной или программной базы; выход из строя сети передачи данных и другие аварийные ситуации.

В основном режиме функционирования система обеспечивает:

- работу пользователей в режиме – 24 часа в день, 7 дней в неделю (24x7);
- выполнение своих функций – сбор, обработка и загрузка данных; хранение данных, предоставление отчетности по показателям.

В профилактическом режиме система обеспечивает возможность проведения следующих работ: - техническое обслуживание;

- модернизацию аппаратно-программного комплекса;
- устранение аварийных ситуаций.

Принимается предварительное решение о том, что общее время проведения профилактических работ не должно превышать X% от общего времени работы системы в основном режиме (XX часов в месяц).

Принимается предварительное решение о том, что для обеспечения высокой надежности функционирования как системы в целом, так и ее отдельных компонентов необходимо проводить регулярное диагностирование состояния компонентов.

В таблице ниже представлены средства диагностики по подсистемам.

Подсистема	Средства диагностирования
Подсистема сбора, обработки и загрузки данных	ETL Administrator – диагностика и настройка ETL-приложения, управление критериями извлечения, установка NLS; ETL Manager - просмотр и редактирование репозитория.
Подсистема хранения данных	DB Manager – диагностика и настройка и конфигурация одной или более БД
Подсистема отображения отчетности	BI Administrator – диагностика и настройка бизнес-описания и представления витрин данных

Далее для каждой подсистемы приводятся примерные сценарии проведения её диагностирования. Чтобы описать сценарии диагностирования необходимо ответить на следующие вопросы: «Кем проводится диагностирование?», «Какое программное обеспечение используется?», «Какие действия (действия прописываются общие, например, зайти, открыть, проверить) необходимо провести для диагностирования?», «Что необходимо проверить? (например, наличие свободного места на дисках)», «Как часто необходимо выполнять данные действия?». Необходимо также указывать критичность подсистемы для функционирования системы в целом.

Например:

Подсистема сбора, обработки и загрузки данных:

- администратор подсистемы должен каждый день контролировать работоспособность серверной части прикладного программного обеспечения сбора, обработки и загрузки данных, т.к. данная подсистема является критичной для работоспособности системы в целом;

- администратор подсистемы перед началом загрузки данных должен проводить контроль объема свободного места на дисках для временных файлов;

- администратор подсистемы должен каждый день проводить анализ протоколов работы подсистемы на наличие ошибок и предупреждений, возникающих при ее работе.

2.4. Решения по персоналу и режимам его работы

На основании пункта «Требования к численности персонала» технического задания приводятся соответствующие решения по численности, квалификации и функциям персонала создаваемой системы, режимам работы персонала.

В данном разделе также формируется таблица с возможными вариантами привязки ролей пользователей и администраторов системы к организационной структуре Заказчика.

Например:

Роль	Подразделение
Конечный пользователь	Аналитическое управление
Администратор подсистемы сбора, обработки и загрузки данных	Департамент информационных технологий
Администратор подсистемы хранения данных	Департамент информационных технологий
...	...

2.5 Сведения об обеспечении заданных в техническом задании потребительских характеристик системы, определяющих ее качество

Приводится таблица трассировки требований, заданных в техническом задании, и описанных проектных решений (достигается, нет, в какой степени, за счет чего?).

Пример:

Требование	Метод реализации
Взаимодействие со смежными системами	Реализуется за счет наличия интерфейсов с системами – источниками данных. Планируется использование промежуточных баз данных; интеграция «точка – точка» (point-to-point); интерактивная загрузка информации из файлов определенного формата.
Диагностирование системы	Реализуется путем определения перечня работ по диагностированию подсистем.
Сохранение работоспособности системы в различных вероятных условиях	Реализуется путем разработки процедур резервного копирования, подготовки персонала, использования современных методов разработки и проверенных на практике стандартных программных средств. На объекте автоматизации обязательно ведение журналов инцидентов в электронной форме, а также графиков и журналов проведения ППР в соответствии с утвержденными для каждого объекта ХД мероприятиями по поддержанию его работоспособности.
...	...

Приводятся сведения по обеспечению заданных в техническом задании требований к функциям, выполняемым каждой подсистемой и определяющим её качество.

Подсистема	Функция	Метод реализации
Подсистема сбора, обработки и загрузки данных	Управление процессами сбора, обработки и загрузки данных	Путем внедрения комплексного ETL-приложения
	Запуск процессов сбора, обработки и загрузки данных из источников в ХД	Путем разработки и внедрения регламентов запуска ETL-процессов

Подсистема хранения данных	Создание и сопровождение структур базы данных	Путем применения CASE-средства и средств администрирования СУБД
	Осуществление резервного копирования данных	Путем применения следующих видов копирования: полное холодное копирование; логическое копирование; инкрементальное копирование

2.6. Состав функций, комплексов задач реализуемых системой

Приводится наименование и назначение функциональных комплексов задач системы (или по каждой подсистеме).

Функциональные задачи по мере проработки проектных решений описываются в виде сценариев. Описания сценариев могут быть вынесены в приложение к пояснительной записке.

Процесс формирования сценариев выполнения каждой задачи функций каждой подсистемы производится следующим образом: приводится наименование подсистемы, наименование функции подсистемы, внутри каждой функции перечисляются задачи, которые выполняются в её рамках (подсистемы, функции, задачи берутся из технического задания), для каждой задачи формируется таблица вида:

Подзадача	Действие
...	...

В данной таблице для каждой задачи приводится перечень подзадач и сценарий их выполнения. Перечень подзадач формируется следующим образом: берется наименование задачи и из названия задачи выделяются подзадачи, например задача «Поддержка (разработка, модификация) модели ХД» содержит в себе две подзадачи «Разработка» и «Модификация», задача

«Создание, редактирование и удаление процессов сбора, обработки и загрузки данных» содержит в себе следующие подзадачи: «Создание нового процесса», «Редактирование процесса», «Удаление процесса» и т.п.

Далее для каждой выделенной подзадачи приводится описание сценариев её выполнения. Сценарий формируется путем последовательных ответов на следующие вопросы:

Вопрос: «Кто производит действия для выполнения подзадачи?»

Ответ: «Администратор подсистемы...»

Вопрос: «Что должен сделать Администратор? К какому ПС обратиться? Какой файл выбрать?»

Ответ: «Администратор подсистемы обращается к программе ... и открывает ранее разработанный ... »

Вопрос: «Какие действия после открытия в рамках подзадачи должен выполнить Администратор?»

Ответ. «Администратор подсистемы обращается к программе ... и открывает ранее разработанный ... Администратор вносит изменения в ..., содержащие ...»

Вопрос: «Какие действия выполняет сама подсистема в момент действия Администратора? Появляется ли диалоговое окно?»

Ответ: «Администратор подсистемы обращается к программе ... и открывает ранее разработанный Администратор вносит изменения в ..., содержащие Подсистема запрашивает необходимость сохранения работы в виде рабочего файла ...»

Вопрос: «Какие действия выполняет Администратор после появления диалогового окна?»

Ответ: «Администратор подсистемы обращается к программе ... и открывает ранее разработанный Администратор вносит изменения в ...,

содержащие Подсистема запрашивает необходимость сохранения работы в виде рабочего файла ... Администратор подтверждает команду сохранения.».

Например, таблица, содержащая описание сценариев для подзадач задачи "Создание, редактирование и удаление процессов сбора, обработки и загрузки данных", функции "Управление процессами сбора, обработки и загрузки данных", подсистемы "Подсистема сбора, обработки и загрузки данных" будет выглядеть следующим образом.

Подзадача	Действие
Создание нового процесса	- Администратор обращается к модулю разработки подсистемы на сервере разработки. - Подсистема предоставляет инструментальные средства для создания нового процесса. - Администратор подсистемы создает схему нового процесса ETL. На схеме указываются компоненты процесса: источники данных, компоненты преобразования данных, таблицы БД. - Администратор подсистемы инициирует команду сохранения созданного процесса. - Подсистема размещает созданный процесс на сервере среды разработки. - Администратор подсистемы выполняет запуск, тестирование и отладку создаваемого процесса. На вход процесса подаются тестовые данные. Анализируя итоговые таблицы БД среды разработки, Администратор принимает решение о готовности нового процесса. - Готовый процесс переносится на продуктивный сервер.
Редактирование процесса	- Администратор подсистемы вызывает подсистему среды разработки на сервере разработки. - Используя инструментальные программные средства подсистемы, Администратор изменяет схему процесса ETL, размещает измененный процесс на сервере среды разработки. - Подсистема размещает редактируемый процесс на сервере среды разработки. - Администратор подсистемы выполняет запуск, тестирование и отладку редактируемого процесса. На вход процесса подаются тестовые данные. Анализируя итоговые таблицы БД среды разработки, Администратор принимает решение о готовности редактируемого процесса. - Готовый процесс переносится на продуктивный сервер.
Удаление процесса	- Администратор подсистемы вызывает подсистему среды разработки на сервере разработки. - Используя инструментальные программные средства подсистемы, Администратор удаляет процесс ETL, размещает изменения на сервере среды разработки. - Подсистема размещает внесенные изменения на сервере среды разработки. - Изменения переносятся на продуктивный сервер.

2.6.1 Подсистема сбора, обработки и загрузки данных

2.6.1.1 Функция «Управление процессами сбора, обработки и загрузки данных»

Описание возможного сценария для последующей реализации задачи «Создание, редактирование и удаление процессов сбора, обработки и загрузки данных» приведено в таблице.

Подзадача	Действие
-----------	----------

Создание нового процесса	- Администратор обращается к модулю разработки подсистемы на сервере разработки.
--------------------------	--

	- Подсистема предоставляет инструментальные средства для создания нового процесса.
--	--

	- Администратор подсистемы создает схему нового процесса ETL. На схеме указываются компоненты процесса: источники данных, компоненты преобразования данных, таблицы БД.
--	---

	- Администратор подсистемы инициирует команду сохранения созданного процесса.
--	---

	- Подсистема размещает созданный процесс на сервере среды разработки.
--	---

	- Администратор подсистемы выполняет запуск, тестирование и отладку создаваемого процесса. На вход процесса подаются тестовые данные. Анализируя итоговые таблицы БД среды разработки, Администратор принимает решение о готовности нового процесса.
--	--

	- Готовый процесс переносится на продуктивный сервер.
--	---

Редактирование процесса	- Администратор подсистемы вызывает подсистему среды разработки на сервере разработки.
-------------------------	--

	- Используя инструментальные программные средства подсистемы, Администратор изменяет схему процесса ETL, размещает измененный процесс на сервере среды разработки.
--	--

	- Подсистема размещает редактируемый процесс на сервере среды разработки.
--	---

- Администратор подсистемы выполняет запуск, тестирование и отладку редактируемого процесса. На вход процесса подаются тестовые данные. Анализируя итоговые таблицы БД среды разработки, Администратор принимает решение о готовности редактируемого процесса.

- Готовый процесс переносится на продуктивный сервер.

Удаление процесса - Администратор подсистемы вызывает подсистему среды разработки на сервере разработки.

- Используя инструментальные программные средства подсистемы, Администратор удаляет процесс ETL, размещает изменения на сервере среды разработки.

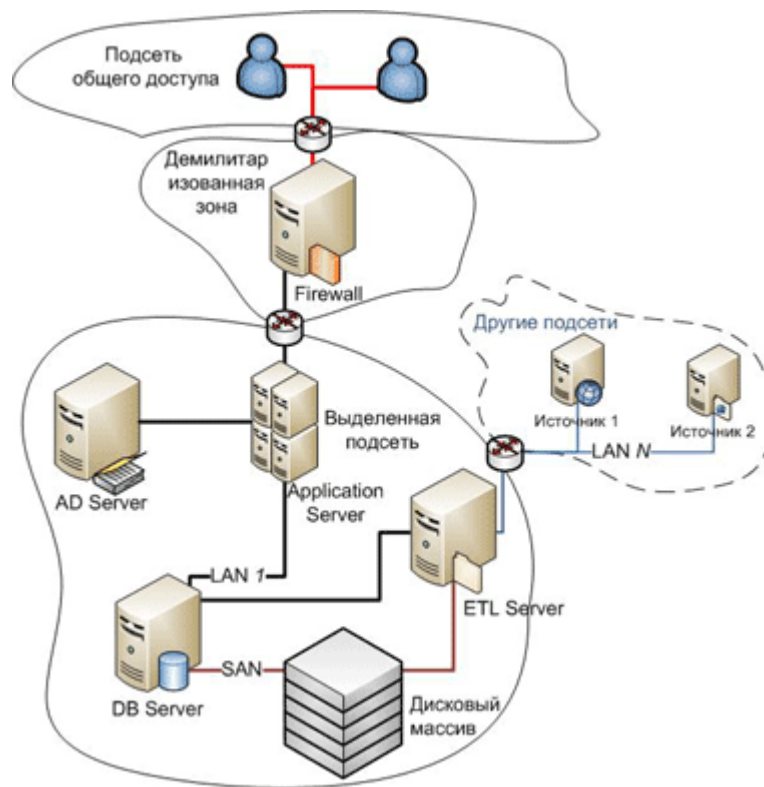
- Подсистема размещает внесенные изменения на сервере среды разработки.

- Изменения переносятся на продуктивный сервер.

2.7. Состав и размещение комплексов технических средств

Решения по комплексу технических средств, его размещению на объекте.

Приводится перечень серверов, рабочих мест, определяется сетевое окружение (включая технические средства), в рамках которого будет функционировать АИС, размещение на технических средствах компонентов.



Например:

Пояснительная записка к эскизному проекту на создание автоматизированной системы - Сетевая архитектура системы

Также определяются предварительные решения по разбивке дискового массива: тома, размеры томов, уровень RAID.

Определяются предварительные решения по резервному копированию: подсистема, тип копирования (холодная копия, логическое копирование, инкрементальное копирование) и его частота, приводятся решения по архивированию копий.

Приводятся предварительные решения по размещению зон разработки, тестирования и промышленной эксплуатации.

2.8. Решения по составу информации, объему, способам ее организации, видам машинных носителей, входным и выходным документам и сообщениям, последовательности обработки информации и другим компонентам

2.8.1 Описание информационной базы

В табличном виде приводится перечень и описание предметных областей модели данных хранилища данных.

Пример:

Предметная область	Описание
Анализ клиентов	В данной области возможен анализ клиентов Заказчика (предприятия, организации и физические лица, потребляющие услуги Заказчика). Например, из данной области можно получить информацию на запросы следующего характера: - Общие запросы по клиентам - Организационно-правовая форма клиента - Месторасположение клиента (страна, город, почтовый адрес) - Контактная информация - Классификация промышленности, к которой относится клиент - Договорные отношения с клиентами - прочее
...	...

Например, из данной области можно получить информацию на запросы следующего характера:

- Общие запросы по клиентам
- Организационно-правовая форма клиента
- Месторасположение клиента (страна, город, почтовый адрес)
- Контактная информация
- Классификация промышленности, к которой относится клиент
- Договорные отношения с клиентами
- прочее
-

Ниже приводятся изображения отношений между сущностями внутри каждой предметной области. Данные изображения формируются на основе концептуальной модели.

соответствующего программного средства, ниже приводится его краткое описание.

Например:

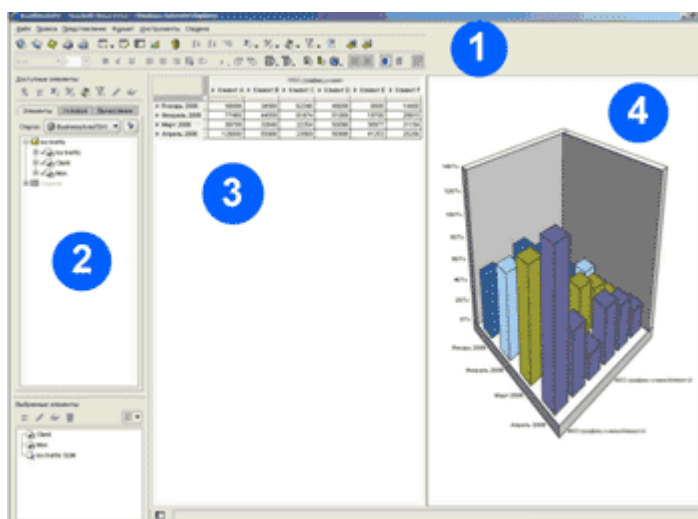
Пример экранной формы вывода анализа данных ВІ средства:

1 – меню, содержащее список команд и панель инструментов.

2 – интерактивное окно редактирования отчета.

3 – таблица с данными.

4 – График, отображающий те же данные, что и в таблице, но в графическом виде.



Пример экранной формы. Пояснительная записка к эскизному проекту

2.9 Методы и средства разработки

Приводятся решения по составу программных средств, языкам деятельности, алгоритмам процедур и операций и методам их реализации.

Данный раздел формируется на основе раздела «Требования к программному обеспечению» технического задания.

Уточнения данного раздела производятся путем ответа на следующие вопросы:

- «Какие программные средства будут использоваться для реализации системы?»»

- «Какие операционные системы будут установлены на серверах?»»

- «Какой язык запросов будет использоваться для работы с БД? В каком стандарте?»»

- «Какие средства будут использоваться для разработки пользовательских интерфейсов и средств генерации отчетов (любых твердых копий)?»»

- «В рамках каких стандартов будут проходить моделирование и описание? С использованием какого программного обеспечения?»»

- «Какие средства и методы разработки программных средств будут использоваться для реализации системы?»».

3. Мероприятия по подготовке объекта автоматизации к вводу системы в действие

В данном разделе указывают:

- мероприятия по приведению информации к виду, пригодному для обработки на ЭВМ;

- мероприятия по обучению и проверке квалификации персонала;

- мероприятия по созданию необходимых подразделений и рабочих мест;

- мероприятия по изменению объекта автоматизации;

- другие мероприятия, исходящие из специфических особенностей, создаваемых АС.

Ниже представлен пример содержания данного раздела.

3.1 Мероприятия по подготовке информационной базы

Приводится перечень мероприятий, которые должны быть проведены в целях приведения информации к виду, пригодному для использования системе КХД. Для этого необходимо ответить на следующий вопрос: «Какие технические решения необходимо согласовать между Разработчиком и Заказчиком?». Например, форматы взаимодействия, способы взаимодействия и т.п.

3.2 Мероприятия по подготовке персонала

Разрабатывается перечень мероприятий, которые необходимо провести Заказчику в целях подготовки пользователей и обслуживающего персонала системы КХД. Например, комплектация штата, назначение ответственных и т.п.

3.3 Мероприятия по организации рабочих мест

Определяется перечень мероприятий, которые должны быть проведены Заказчиком в целях организации рабочих мест разработчиков, пользователей, администраторов системы. Например, организовать подсеть разработчиков и администраторов, организовать обучение и т.п. Также в этом разделе приводятся предварительные требования к рабочим местам. Например, указывается, что на рабочих станциях пользователей должен быть установлен MS Internet Explorer не ниже версии 5.5 и т.п.

3.4 Мероприятия по изменению объекта автоматизации

Приводится перечень мероприятий, которые должны быть проведены силами Заказчика в целях подготовки помещений для размещения аппаратно-технического комплекса системы и организации необходимого аппаратно-технического обеспечения. Например, организовать сетевое взаимодействие, закупить оборудование и т.п.

3.5 Прочие мероприятия

Указываются мероприятия по изменению объекта автоматизации, другие мероприятия, исходящие из специфических особенностей создаваемой АИС.

[Ковтун М.В., Байбородов К.М. Август 2010. https://www.prj-exp.ru/patterns/pattern_draft_project.php]

Работа 3 функциональное проектирование модели информационной системы системы с использованием методологии SADT

Создание краткого описания и контекстной диаграммы проектируемой системы в нотации IDEF0. Задание: создание краткого описания объекта автоматизации, постановка задачи процесса автоматизации, описание основных параметров проектируемой информационной системы, описание путей достижения целей. Создать контекстную диаграмму в нотации IDEF0.

Задание: создание краткого описания объекта автоматизации, постановка задачи процесса автоматизации, описание основных параметров проектируемой информационной системы, описание путей достижения целей. декомпозировать функциональную модель проектируемой системы в нотации IDEF0, создать декомпозицию, как минимум два уровня.

Допустимо декомпозировать один, наиболее значимый для проектируемой системы, функциональный блок на каждом уровне.

Цель работы

Целью данной практической работы является выбор и проектирование информационной системы, составление её краткого описания, включая цель создания ИС, способ и средства создания, а также моделирование контекстной диаграммы A-0 в нотации IDEF0.

Ход работы

Каждый студент должен выбрать объект автоматизации в качестве базы для реализации программного комплекса. Варианты объектов для автоматизации даны в приложении А. Кроме того, по согласованию с руководителем студент вправе предложить собственный тип объекта автоматизации. В случае проектирования сложных аппаратно-программных комплексов, которые невозможно целиком реализовать за время изучения

дисциплины, допускается бригадная работа студентов, при условии, что каждый студент будет выполнять проектирование собственного модуля ИС.

Работу следует оформлять в соответствии с ГОСТ 7.32–2017 Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления" (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) и СМКО МИРЭА. Для проектирования контекстной диаграммы функциональной схемы ИС в нотации IDEF0 необходимо использовать РД IDEF0 – 2000. Методология функционального моделирования IDEF0 Руководящий документ.

Пример выполнения работы:

Для проектирования была выбрана информационная система автоматизации поиска интересующих потребителя фильмов, обсуждения темы кинофильмов с другими пользователями, а также возможности приобретения билетов в кинотеатр своего города. Название системы «StudioHD». Система создается для нужд сервиса он-лайн продаж билетов в кинотеатры.

Цель создания ИС

Целью создания ИС «StudioHD» является увеличение спроса на приобретение билетов в кинотеатры с перспективой на окупаемость путем внедрения реферальной программы, создание платформы с большим количеством пользователей, где впоследствии различные рекламодатели смогут размещать нативную рекламу с целью получения прибыли.

По определению ИС: «Информационная система – это сложный программный комплекс, который способен собирать, сохранять, обрабатывать и выдавать по запросу пользователя информацию». Проектируемая ИС полностью удовлетворяет всему перечню требования, указанные в определении, т.к.:

1) Сайт собирает информацию о предпочтениях у потребителей определенных жанров фильмов, актеров, режиссеров и другой информации,

которая поможет пользователю выбрать интересный для него сеанс. Также системой собираются персональные данные участников платформы. Помимо этого, платформа собирает информацию о киносеансах конкретного города и регулярно её обновляет;

2) Хранит полученную информацию в базе данных;

3) Информация из подпунктов выше обрабатывается, на основе чего при помощи специальных алгоритмов формируются рекомендации к посещению определенных киносеансов;

4) Доступ пользователей к огромному количеству информации на сайте (биографии актеров, факты о съемках фильмов, рейтинги и т.д.);

Краткое описание

ИС «StudioHD» представлена в виде сайта. Сайт является удобным интернет сервисом, представляющим информацию о кинофильмах, актерах режиссерах и других связанных с фильмами тематиках. Для комфортного и круглосуточного доступа, сайт так же адаптирован для мобильных устройств.

Одним из важных достоинств проектируемой ИС – большой функционал для взаимодействия с личным профилем клиента. Особо активные участники смогут выставлять оценки фильмам и писать о них отзывы. Это даст дополнительный интерес к платформе активным пользователям, что положительно скажется на посещаемости ресурса.

Кроме того, платформа позволяет пользователю просматривать билеты на интересующий его киносеанс, при условии того, что кинотеатры его города приняли правила платформы и подключены к системе.

Способ создания ИС

В качестве способа определения требований была выбрана методология «последовательных приближений», которая основана на том, что все расчеты и графические построения, связанные с определением основных элементов, разбиваются на несколько более мелких элементов, в которых происходит их уточнение. Данный метод также хорошо сочетается с нотацией IDEF0, которая

основана на декомпозиции каждого блока на более мелких с уточнением деталей.

Средства создания ИС

В качестве средств создания ИС был использован язык программирования Java (фреймворк Spring), СУБД MySQL и сервис для развёртывания сервера nginx. Для моделирования проектируемой ИС будет использоваться нотация IDEF0 программном обеспечении CASE Ramus Educational edition.

Проектирование контекстной диаграммы функциональной модели ИС

Была спроектирована контекстная диаграмма А–0 в нотации IDEF0.

В качестве входа по управлению (стрелка управления) были выбраны следующие нормативные и правовые документы:

- 1) Закон о персональных данных;
- 2) Политика сайта;
- 3) Алгоритмы для обработки информации о предпочтениях пользователей и выдачи рекомендаций;

В качестве входящих информационных потоков, которые подлежат обработке и преобразованию в процессе работы ИС была указана следующая информация:

- 1) Персональные данные пользователя;
- 2) Предпочтения пользователей;

В качестве механизмов (ресурсов, выполняющих работу) были выделены:

- 1) Модуль обработки данных профиля пользователей;
- 2) Пользователь, взаимодействующий с системой по продаже билетов;
- 3) Пользователь сайта;

В качестве выходов после выполнения ИС получены следующие информационные элементы:

- 1) Статистические данные;

2) Информация о покупке билета;

На рисунке 1.1 представлена контекстная диаграмма проектируемой информационной системы. Более подробная информация по входам и управляющим воздействиям, с подробным описанием, представлена в Глоссарии.

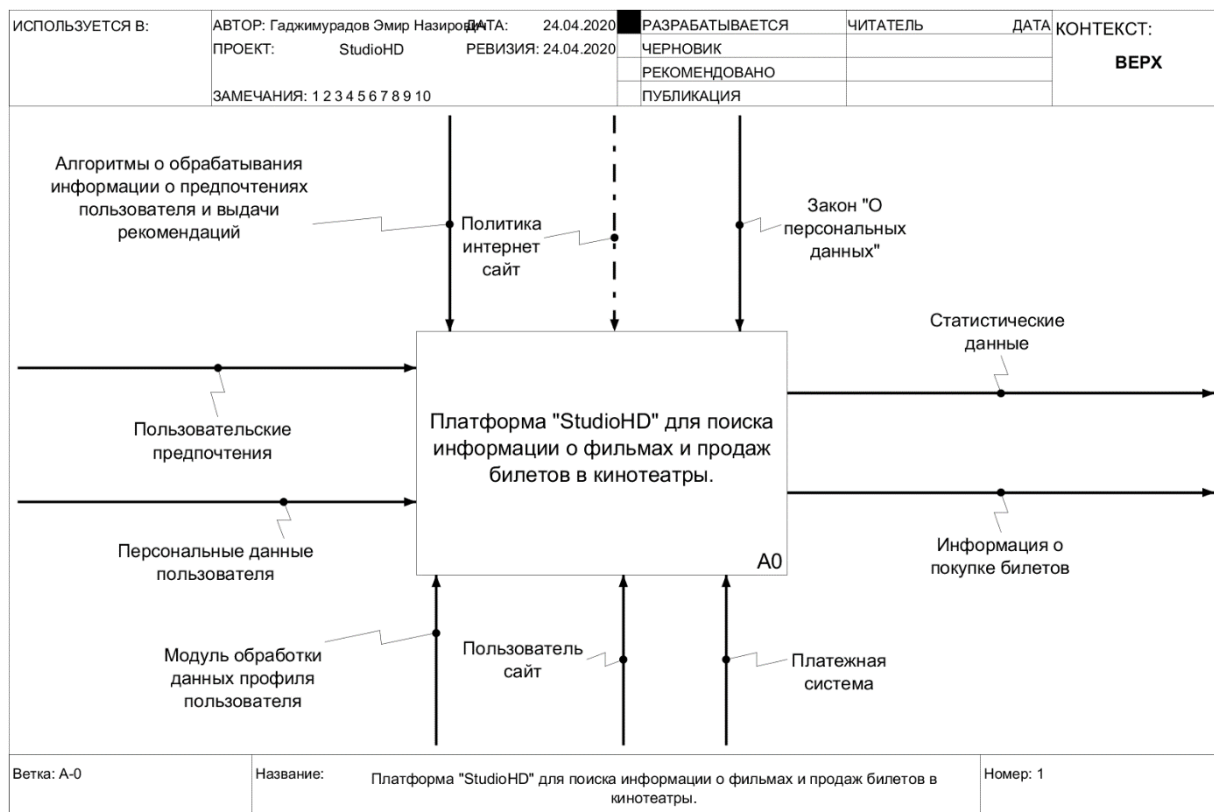


Рисунок 1.1 – Контекстная диаграмма процесса управления файлами в ИС онлайн продаж билетов «Studio HD»

Вывод

Итогом работы стала выбранная информационная система, определена цель, способ и средства создания ИС, составлено краткое описание, а также смоделирована контекстная диаграмма A-0 в нотации IDEF0.

Список использованных источников

- 1) Исаев, Г.Н. Проектирование информационных систем: Учебное пособие / Г.Н. Исаев.— М.: Омега-Л, 2013. – 424 с.
- 2) Коваленко, В.В. Проектирование информационных систем: Учебное пособие / В.В. Коваленко.— М.: Форум, 2015. – 976 с.

3) Соловьев, И.В. Проектирование информационных систем. Фундаментальный курс: Учебное пособие для высшей школы / И.В. Соловьев, А.А. Майоров; Под ред. В.П. Савиных.— М.: Академический проспект, 2009. – 398 с.

4) Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий / Н.В. Федоров.— М.: МГИУ, 2008. – 280 с.

Работа 4 Проектирование функциональной модели информационной системы в нотации IDEF0

Задание: декомпонировать функциональную модель проектируемой системы в нотации IDEF0, создать декомпозицию, как минимум два уровня. Допустимо декомпонировать один, наиболее значимый для проектируемой системы, функциональный блок на каждом уровне.

Цель работы

Целью данной практической работы является продолжения проектирования выбранной информационной системы, моделирования (минимум два уровня декомпозиции) двух уровней декомпозиции А-1, А-2 в нотации IDEF0 и составить текстовое описание проектируемых модулей на уровнях декомпозиции. Работу следует оформлять в соответствии с ГОСТ 7.32–2017 Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления" (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) и СМКО МИРЭА. Для проектирования контекстной диаграммы функциональной схемы ИС в нотации IDEF0 необходимо использовать РД IDEF0 – 2000. Методология функционального моделирования IDEF0 Руководящий документ.

Ход работы

Используя контекстную диаграмму проекта информационной системы, полученную по итогам выполнения предыдущей практической работы, выбрать и декомпонировать как минимум один блок. Декомпозируемый блок необходимо согласовать с преподавателем.

Пример выполнения работы:

На диаграмме уровня А0 декомпозиции функционального блока «Платформа “StudioHD” для поиска информации о фильмах и продажи

билетов в кинотеатры» обозначены процессы и функциональные блоки, выполняемые в рамках процедуры:

- Регистрация пользователя в системе (A1);
- Процесс приобретения пользователем билета в кино(A2);
- Получение билета и чека пользователем (A3).

Итак, первый процесс который происходит с момента посещения пользователем сайта– это «Регистрация пользователя в системе». Каждый пользователь, который хочет полноценно использовать данный ресурс должен быть зарегистрирован в системе. В качестве исходных данных функциональный блок принимает в себя персональные данные пользователя и его вкусовые предпочтение в сфере кино для построения списка рекомендаций. Процесс регистрации проходит согласно закону “О персональных данных” и политике интернет - сайта. После регистрации пользователя и выбора этим пользователем понравившегося фильма, происходит перенаправление на следующий процесс – «Процесс приобретения пользователем билета в кино» (рисунок 1.2).

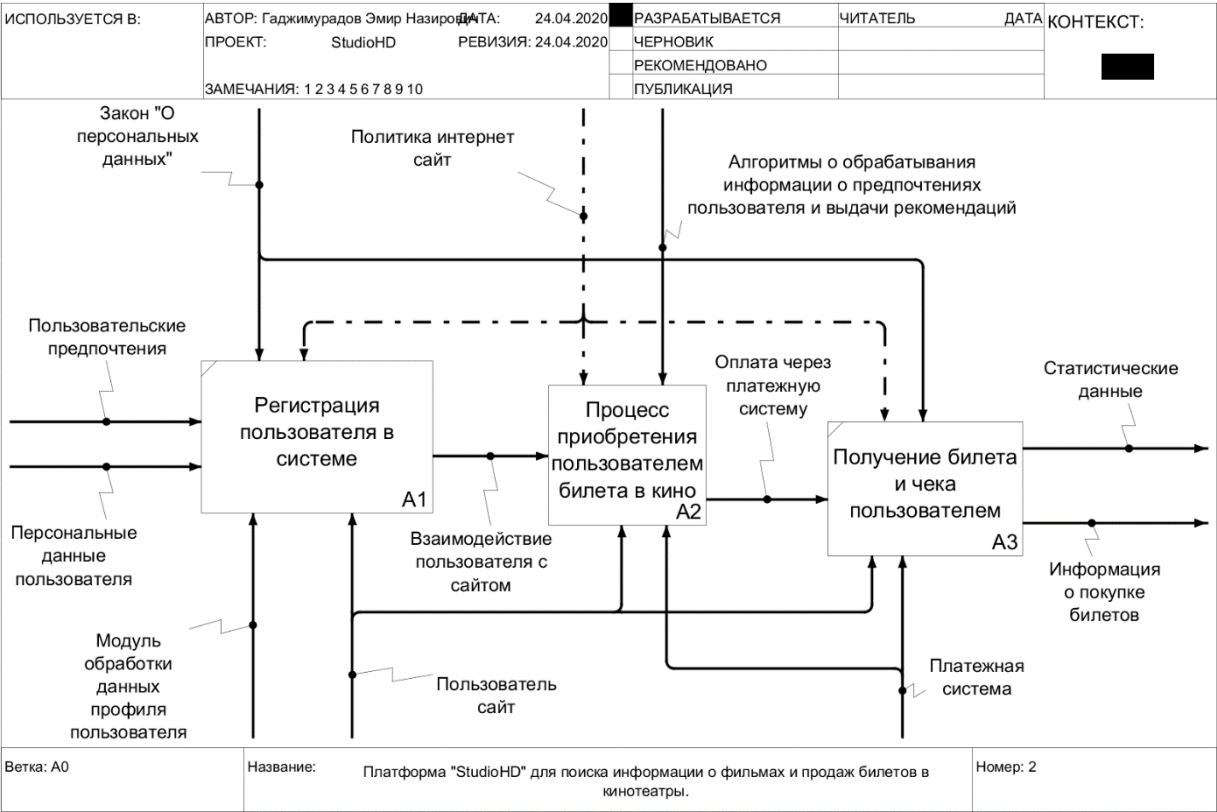


Рисунок 1.2 – Диаграмма декомпозиции блока «Платформа “StudioHD” для поиска информации о фильмах и продажи билетов в кинотеатры» в нотации IDEF0.

На диаграмме уровня A0 декомпозиции функционального блока «Платформа “StudioHD” для поиска информации о фильмах и продажи билетов в кинотеатры» обозначены процессы и функциональные блоки, выполняемые в рамках процедуры:

- Регистрация пользователя в системе (A1);
- Процесс приобретения пользователем билета в кино(A2);
- Получение билета и чека пользователем (A3).

Итак, первый процесс который происходит с момента посещения пользователем сайта– это «Регистрация пользователя в системе». Каждый пользователь, который хочет полноценно использовать данный ресурс должен быть зарегистрирован в системе. В качестве исходных данных функциональный блок принимает в себя персональные данные пользователя и его вкусовые предпочтения в сфере кино для построения списка рекомендаций. Процесс регистрации проходит согласно закону “О персональных данных” и политике интернет - сайта. После регистрации пользователя и выбора этим пользователем понравившегося фильма, происходит перенаправление на следующий процесс – «Процесс приобретения пользователем билета в кино».

Функциональный блок «Процесс приобретения пользователем билета в кино» раскрывает процесс покупки пользователем билета, выгружая необходимые сведения в шаблон. На вход поступает выбранный пользователем фильм, на который можно купить билет, после чего на выходе пользователь оплачивает билет через платежную систему.

Блок «Получение билета и чека пользователем» отвечает за выдачу чека о успешном платеже и билета для пользователя. Этот процесс так же

происходят в соответствии с политикой сайта и законом “О персональных данных”. На выходе получаем электронный билет в кино.

Рассмотрим диаграмму процессов, происходящих в функциональном блоке А2, приведенном выше.

На рисунке 1.3 рассмотрена декомпозиция функционального блока А2. Исходя из детального уточнения выполняемых задач ИС, были определены следующие функциональные элементы:

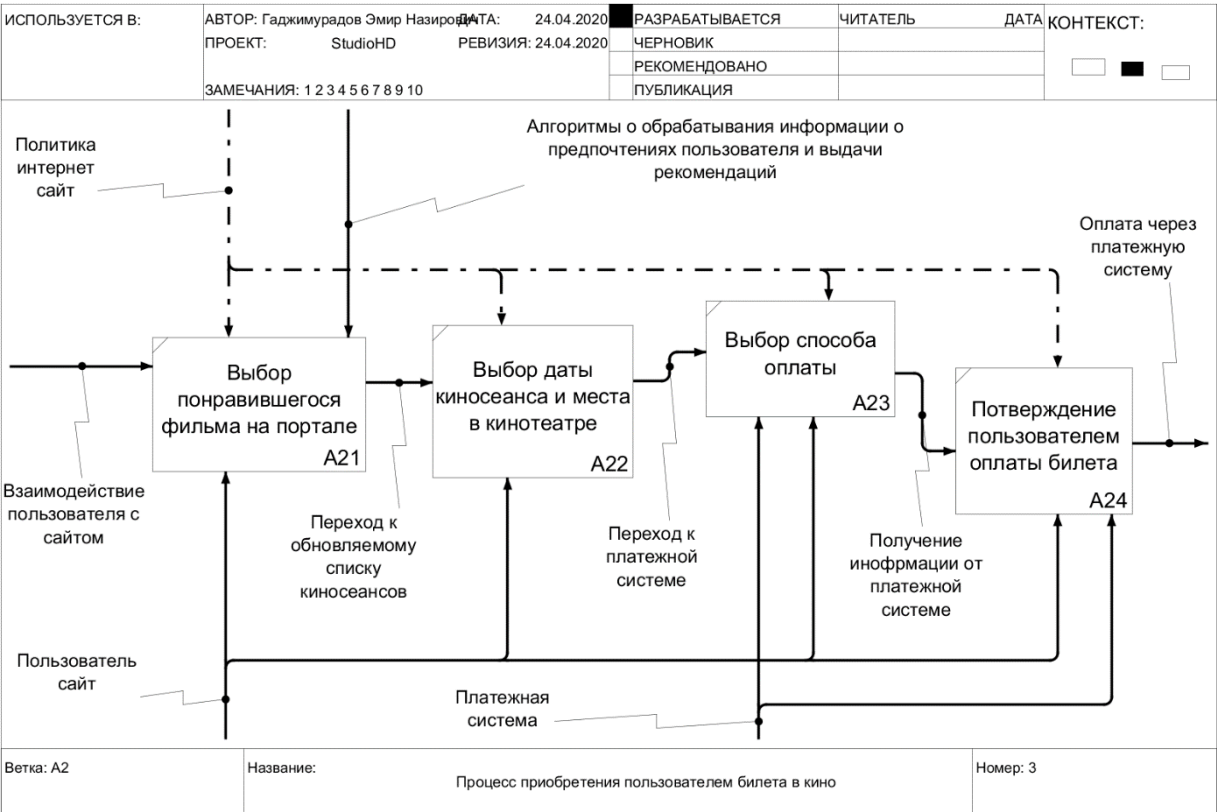


Рисунок 1.3 – Диаграмма декомпозиции функционального блока А2

- Выбор понравившегося фильма на портале (A21);
- Выбор даты киносеанса и места в кинотеатре (A22);
- Выбор способа оплаты(A23);
- Подтверждение пользователем оплаты билета(A24).

Первый процесс, протекающий на диаграмме декомпозиции А2 – это «Выбор понравившегося фильма на портале». Сперва, когда человек хочет пойти в кинотеатр, он выбирает фильм основываясь на своих вкусовых

предпочтениях. Следующим, является блок «Выбор даты киносеанса и места в кинотеатре». Пользователя перенаправляют на страницу, где он на основе выбранной им даты может выбрать для себя место в кинозале. В функциональном блоке «Выбор способа оплаты», пользователь перенаправляется в платежную систему, которая подключена к ИС для выбора способа оплаты. Функциональный блок «Подтверждение пользователем оплаты билета» берет все нижеописанные данные (дата сеанса, место в кинотеатре, способ оплаты) и выведет финальное окно, где пользователь должен подтвердить правильность введенных им данных, только после этого оплатив билет, процесс получения которого будет реализован в блоке А3.

Вывод

Во время выполнения работы были смоделированы два уровня декомпозиции в нотации IDEF0, составлено текстовое описание проектируемых модулей и функций программного комплекса на двух уровнях декомпозиции и подробно описан алгоритм выдачи квалификации участников тестирования на втором уровне декомпозиции.

Список использованных источников

- 1) Исаев, Г.Н. Проектирование информационных систем: Учебное пособие / Г.Н. Исаев.— М.: Омега-Л, 2013. – 424 с.
- 2) Коваленко, В.В. Проектирование информационных систем: Учебное пособие / В.В. Коваленко.— М.: Форум, 2015. – 976 с.
- 3) Соловьев, И.В. Проектирование информационных систем. Фундаментальный курс: Учебное пособие для высшей школы / И.В. Соловьев, А.А. Майоров; Под ред. В.П. Савиных.— М.: Академический проспект, 2009. – 398 с.
- 4) Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий / Н.В. Федоров.— М.: МГИУ, 2008. – 280 с.

Работа 4 проектирование модели потоков данных в нотации DFD.

Декомпозиция функционального блока (функциональных блоков) в нотации DFD. Задание: выбрать наиболее значимый функциональный блок нижнего уровня декомпозиции из предыдущей лабораторной работы и выполнить его декомпозицию в нотации DFD. Декомпозиция выполняется как отдельный файл в нотации диаграммы потоков данных. Допустимо выполнение одноуровневого описания потоков данных, однако рекомендуется двухуровневое описание. Вопрос одного или двух уровней декомпозиции в нотации DFD рекомендуется согласовать с преподавателем, ведущим занятия.

Цель работы

Цель работы заключается в выборе наиболее значимого функционального блока нижнего уровня декомпозиции из предыдущей работы и выполнения его декомпозицию в нотации DFD.

Целью данной работы является продолжение создания описания проектируемой информационной системы, а также создание следующей уровня декомпозиции в нотации DFD (диаграмма потоков данных) и подготовке к выполнению диаграммы сущность – связь проектирования баз данных ИС.

Ход работы

Для декомпозиции в нотации DFD проектируемой системы выбрать один из блоков декомпозиции, полученных в предыдущей работе и дальнейшая декомпозиция (не менее двух уровней) в нотации DFD. Работу следует оформлять в соответствии с ГОСТ 7.32–2017 Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления" (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) и СМКО МИРЭА. Для проектирования контекстной

диаграммы функциональной схемы ИС в нотации IDEF0 необходимо использовать РД IDEF0 – 2000. Методология функционального моделирования IDEF0 Руководящий документ.

Пример выполнения работы:

«Платформа “StudioHD” для поиска информации о фильмах и продажи билетов в кинотеатры» был выбран блок второго уровня декомпозиции ветки А2 с номером 2 «Выбор даты киносеанса и места в кинотеатре» (блок А22). Во время этого этапа выполняется выбор пользователем места в кинотеатре, а также обработка этого выбора системой с последующим подтверждением от пользователя.

Данный этап является достаточно важным, потому что посетителям кинотеатра нужно с комфортом наслаждаться просмотром кинофильма, а выбор хорошего места, основываясь на личном предпочтении, является одной из частиц, которые позволят будущему посетителю приятно провести своё время отдыха.

Полученная схема в нотации DFD изображена на рисунке 4.

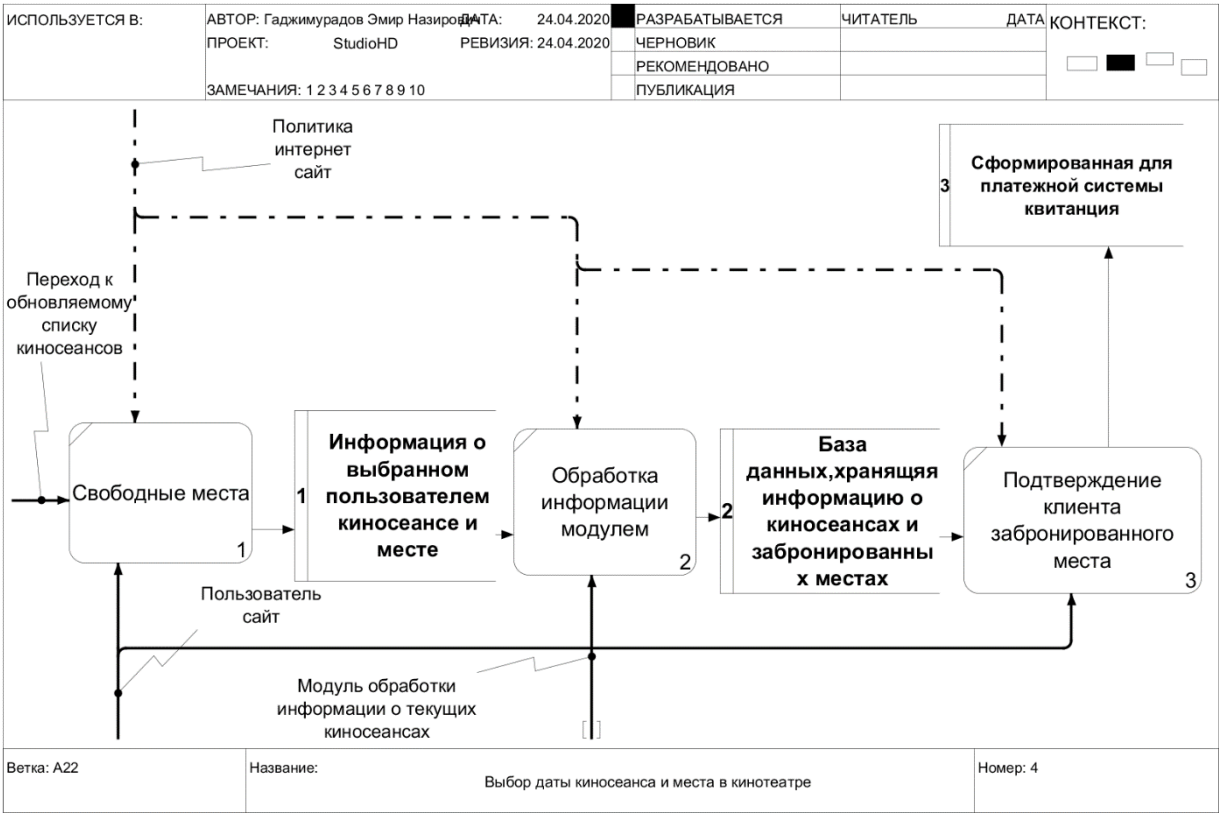


Рисунок 4 – Диаграмма «Выбор даты киносеанса и места в кинотеатре» в нотации DFD

Было выделено три основных процесса на диаграмме потоков данных. Первый из них – это «Выбор пользователем свободного места». Второй – «Обработка информации модулем», третий – «Подтверждение пользователем забронированного места».

Процесс «Выбор пользователем свободного места»

На вход у нас поступает «Обновляемый список киносеансов», рассматривая который пользователь выбирает заинтересовавший его киносеанс. В качестве стрелки управления поступает «Политика интернет сайта». В качестве стрелки механизмов поступает «Пользователь сайта» и «Модуль обработки информации о текущих киносеансах». А на выход поступает хранилище данных «Информация о выбранном пользователем месте».

Процесс «Обработка информации модулем»

В данном процессе система получает место от пользователя, чтобы отобразить его как забронированное и внести его в обновляемую базу данных.

На входе мы получаем данных из хранилища «Информация о выбранном пользователем киносеансе и месте. В качестве стрелки управления поступает «Политика интернет сайта». В качестве стрелки механизмов поступает «Модуль обработки информации о текущих киносеансах». А на выход хранилище данных «База данных, хранящая информацию о киносеансах и забронированных местах».

Процесс «Подтверждение пользователем забронированного места»

Последний процесс, позволяющий после подтверждения пользователем правильности введенной информации перейти в платежную систему для оплаты билета.

На вход так же используется хранилище из предыдущего этапа, а именно «Выбранное пользователем свободное место». В качестве стрелки управления

поступает «Политика интернет сайта». В качестве стрелки механизмов поступает «Пользователь сайта».

На выходе процесс выдаёт хранилище «Сформированная для платежной системы квитанция» для дальнейшей оплаты пользователем билета, что является промежуточным итоговым результатом всей работы схемы, т.к. при определённых обстоятельствах пользователь может отказаться от похода в кинотеатр.

Вывод

Во время выполнения работы было продолжение составления описание алгоритмов и работы, проектируемой ИС, а также создан следующий уровень декомпозиции в нотации DFD.

Список использованных источников

- 1) Исаев, Г.Н. Проектирование информационных систем: Учебное пособие / Г.Н. Исаев.— М.: Омега-Л, 2013. – 424 с.
- 2) Коваленко, В.В. Проектирование информационных систем: Учебное пособие / В.В. Коваленко.— М.: Форум, 2015. – 976 с.
- 3) Соловьев, И.В. Проектирование информационных систем. Фундаментальный курс: Учебное пособие для высшей школы / И.В. Соловьев, А.А. Майоров; Под ред. В.П. Савиных.— М.: Академический проспект, 2009. – 398 с.
- 4) Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий / Н.В. Федоров.— М.: МГИУ, 2008. – 280 с.

Работа 5 проектирование структуры данных информационной системы и создание ER-диаграммы

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Описание лабораторного практикума включает в себя учебно-методические материалы к выполнению семи лабораторных работ по всем темам рабочей программы дисциплины «Проектирование информационных систем».

Работа выполняется как во время аудиторных занятий, так и в виде самостоятельной внеаудиторной работы. Выполнение каждой лабораторной работы состоит из трёх этапов:

1. Подготовка и получение допуска к работе.
2. Получение индивидуального задания и выполнение основной части работы.
3. Оформление и защита отчёта о проделанной работе.

В начале каждой лабораторной работы выполняется повторение теоретического материала и проверка готовности к выполнению работы с помощью контрольных вопросов. После получения допуска к выполнению работы выдаётся индивидуальный вариант задания для самостоятельной работы. На заключительном этапе оформляется отчёт о проделанной работе с описанием полученных результатов и выполняется процедура защиты отчёта.

Процедура защиты отчёта заключается в проверке:

- 1) правильности структуры, содержания и оформления отчёта;
- 2) корректности полученных результатов и полноты их описания;
- 3) способности дать объяснение и необходимое обоснование полученным результатам.

Отчет должен включать в себя:

1. Титульный лист.

2. Задание на лабораторную работу.
3. Содержание отчёта.
4. Описание результатов по каждой части задания.
5. Приложение (диаграммы *UML*, тексты программ, содержание проектных документов и т.д.).

Цели и задачи работы

Целями выполнения лабораторной работы являются:

1. Закрепление имеющихся знаний о базах данных. Изучение методологии проектирования базы данных как основы информационной системы.
2. Приобретение навыков анализа и формализованного описания заданной предметной области.
3. Приобретение навыков разработки проекта базы данных с учётом её использования в составе некоторой информационной системы.

В процессе выполнения лабораторной работы решаются следующие задачи:

1. Выполняется системный анализ заданной предметной области. Составляется формализованное описание информационных объектов предметной области.
2. Разрабатывается концептуальная модель базы данных, описывающая сущности предметной области и связи между ними.
3. Выполняется логическое проектирование реляционной базы данных. Составляются типовые запросы на языке *SQL* для поиска и анализа информации.

Краткие теоретические сведения

База данных (БД) – это совокупность данных, отображающая состояние объектов и их отношения в рассматриваемой предметной области. База данных является основой любой информационной системы.

Модель данных – это некоторая абстракция, которая в приложении к конкретным данным позволяет пользователям и разработчикам

трактовать их как информацию, т. е. рассматривать их как сведения, содержащие не только данные, но и взаимосвязи между ними.

Реляционная модель данных основана на понятии *отношения*, физическим представлением которого является двухмерная таблица, состоящая из строк одинаковой структуры. Логическая структура данных представляется набором связанных таблиц.

Система управления базами данных (СУБД) – это совокупность лингвистических и программных средств, необходимых для создания и использования БД. СУБД предоставляют прикладным программам, разработчикам и пользователям множество различных представлений данных, хранящихся в БД.

Порядок выполнения работы

Вариант индивидуального задания определяет предметную область для разработки проекта базы данных некоторой информационной системы.

В процессе выполнения лабораторной работы необходимо:

1. Составить план разработки проекта базы данных для заданной предметной области. Базу данных следует рассматривать как часть будущей информационной системы, автоматизирующей бизнес-процессы некоторой организации.

2. Выполнить анализ заданной предметной области. Сформулировать словесное описание информационных объектов. Описать типовые запросы для поиска и анализа информации об объектах предметной области.

3. Построить концептуальную модель данных, описывающую предметную область в рамках *ER*-модели «сущность – связь». Получить визуальное представление концептуальной модели путём построения *ER*-диаграмм.

4. Построить логическую модель базы данных. Преобразовать полученные ранее *ER*-модели в конкретную схему реляционной базы данных.

5. Проверить полноту и корректность логической модели базы данных путём составления на языке *SQL* типовых запросов для поиска и анализа информации.

6. Модели, полученные на этапах анализа предметной области, концептуального и логического проектирования, а также результаты составления и проверки типовых запросов оформить в виде общего документа – проекта базы данных.

Варианты индивидуальных заданий

Варианты индивидуальных заданий взять из работ №1.

Правила построения диаграмм сущность-связь

Важнейшим компонентом любой информационной системы является **База данных (БД)**. **База данных (Data Base)** – структурированный, организованный набор данных, объединенный в соответствии с некоторой выбранной моделью и описывающий характеристики какой-либо физической или виртуальной системы.

Именно **БД** позволяет эксплуатировать **ИС**, выполнять ее текущее обслуживание, модифицировать и развивать её при модернизации предприятия (организации) или изменении информационных потоков, законодательства и форм отчетности предприятия (организации).

Согласно современной методологии, процесс создания **ИС** представляет собой процесс построения и последовательного преобразования ряда согласованных **моделей** на всех этапах жизненного цикла (**ЖЦ**) **ИС**. На каждом этапе **ЖЦ** создаются **модели**: организации, требований к **ИС**, проекта **ИС**, требований к приложениям и т. д.

Проектирование ИС охватывает три основные области:

- проектирование объектов данных (**создание моделей данных**), которые будут реализованы в базе данных;
- проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
- учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т. п.

Модель – искусственный объект, представляющий собой отображение (образ) системы и её компонентов.

Модель данных (Data Model) – это графическое или текстовое представление анализа, который выявляет данные, необходимые организации с целью достижения ее миссии, функций, целей, стратегий, для управления и оценки деятельности организации. **Модель**

данных выявляет **сущности, домены (атрибуты)** и **связи** с другими данными, а также предоставляет концептуальное представление данных и связи между данными.

Цель создания **модели данных** состоит в обеспечении разработчика **ИС концептуальной** схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть интегрированы в любую базу данных.

При создании моделей данных используется метод **семантического моделирования**. Семантическое моделирование основывается на значении структурных компонентов или характеристик данных, что способствует правильности их интерпретации (понимания, разъяснения). В качестве инструмента семантического моделирования используются различные варианты **диаграмм сущность-связь (ER — Entity-Relationship) — ERD**.

Существуют различные варианты отображения **ERD**, но все варианты **диаграмм сущность-связь** исходят из одной идеи — рисунок всегда нагляднее текстового описания. **ER -диаграммы** используют графическое изображение **сущностей** предметной области, их свойств (**атрибутов**), и **взаимосвязей** между сущностями.

Базовые понятия ERD

Сущность (таблица, отношение) — это представление набора реальных или абстрактных объектов (людей, вещей, мест, событий, идей, комбинаций и т. д.), которые можно выделить в одну группу, потому что они имеют одинаковые характеристики и могут принимать участие в похожих связях. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность в модели изображается в виде прямоугольника с наименованием.

Можно сказать, что **Сущности** представляют собой множество реальных или абстрактных вещей (людей, объектов, событий, идей и т. д.), которые имеют общие **атрибуты** или характеристики.

Экземпляр сущности (запись, кортеж)- это конкретный представитель данной сущности.

Атрибут сущности (поле, домен) — это именованная характеристика, являющаяся некоторым свойством сущности.

Связь — это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Каждая **связь** может иметь один из следующих **типов связи**:

Один-к-одному, многое-ко-многим, один-ко-многим.

Связь типа **один-к-одному** означает, что **один экземпляр первой сущности** (левой) связан с **одним экземпляром второй сущности** (правой). Связь **один-к-одному** чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа **многое-ко-многим** означает, что **каждый экземпляр первой сущности** может быть связан с **несколькими экземплярами второй сущности**, и **каждый экземпляр второй сущности** может быть связан с **несколькими экземплярами первой сущности**. Тип связи **много-ко-многим** является **временным** типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа **один-ко-многим** путем создания промежуточной сущности.

Связь типа **один-ко-многим** означает, что **один экземпляр первой сущности** (левой) связан с **несколькими экземплярами второй сущности** (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны «один») называется **родительской**, правая (со стороны «много») — **дочерней**.

При разработке **ER-моделей** необходимо обследовать предметную область (организацию, предприятие) и выявить:

- 1) Сущности, о которых хранятся данные в организации (предприятии), например, люди, места, идеи, события и т.д., (будут представлены в виде блоков);

- 2) Связи между этими сущностями (будут представлены в виде линий, соединяющих эти блоки);
- 3) Свойства этих сущностей (будут представлены в виде имен атрибутов в этих блоках).

Порядок выполнения работы

Задача: разработать информационную систему «**Контингент студентов института**».

Необходимо: изучить предметную область (образовательное учреждение) и процессы, происходящие в ней.

Для этого обследуем объект: знакомимся с нормативной документацией, опрашиваем работников института, изучаем существующий документооборот института, анализируем ситуацию и т.п.

В результате обследования определяем **цель и задачи системы** и формулируем постановку задачи.

Краткая постановка задачи: главная задача системы – сбор и обработка информации об основных участниках учебного процесса: студентах и преподавателях, формирование необходимых печатных форм (документов), используемых преподавателями в период зачётной недели и экзаменационной сессии, генерация сводных отчётов по результатам сессии для работников деканатов, института. При разработке системы следует учитывать, что она основывается на документации, поступающей из приёмной комиссии, деканатов и других подразделений института. Информация об успеваемости студентов должна накапливаться и храниться в течение всего периода обучения. В системе должен использоваться справочник специальностей и дисциплин (предметов), изучаемых студентами.

Таким образом, проектируемая система должна выполнять следующие действия:

- 1) Хранить информацию о студентах и их успеваемости.
- 2) На факультетах по определённой специальности печатать экзаменационные ведомости и другие документы.

- 3) Выделим все существительные в этих предложениях — это предполагаемые **сущности** и проанализируем их:
- 4) **Студент** — явная сущность.
- 5) **Успеваемость** — явная сущность.
- 6) **Факультет** — нужно выяснить один или несколько факультетов в институте? Если несколько, то это — предполагаемая новая сущность.
- 7) **Специальность** — нужно выяснить одна или несколько специальностей на факультете? Если несколько, то это — ещё одна сущность.
- 8) **Предмет** — предполагаемая сущность.
- 9) На первоначальном этапе моделирования данных информационной системы явно выделены две основные сущности: **Студент** и **Успеваемость**.
- 10) Критерием успеваемости является наличие отметки о сдачи экзаменов.
- 11) Сразу возникает очевидная связь между сущностями — «студент сдаёт несколько экзаменов» и «экзамены сдаются каждым студентом». Явная связь **Один-ко-многим**. Первый вариант диаграммы выглядит так:

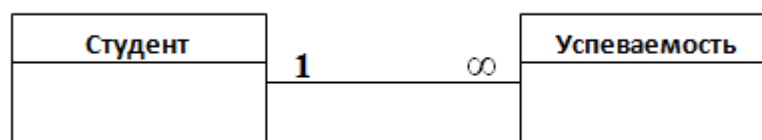


Рисунок 5.1 – Пример связи один ко многим

Мы знаем, что студенты учатся на факультетах, на определённой специальности и сдают экзамены по дисциплинам (предметам). Анализ предметной области показал, что студенты учатся на нескольких факультетах института по нескольким специальностям и сдают экзамены по определённому перечню предметов.

Исходя из этого, мы добавляем в **ER-модель** ещё несколько сущностей. В результате она будет выглядеть так:



Рисунок 5.2 – Пример построения ER-диаграммы

На следующей стадии проектирования модели вносим атрибуты сущностей в диаграмму (предполагаем, что атрибуты выявлены на стадии обследования объекта и при анализе аналогов существующих систем) и получаем окончательный вариант **ER— диаграммы**:



Рисунок 5.3 –ER-диаграмма дополненная атрибутами сущностей

Отметим, что предложенные этапы моделирования являются условными и нацелены на формирование общих представлений о процессе моделирования.

Разработанный выше пример **ER-диаграммы** является примером **концептуальной диаграммы**, не учитывающей особенности конкретной СУБД. На основе данной концептуальной диаграммы можно построить **физическую диаграмму**, которая будут учитывать такие особенности СУБД, как допустимые типы, наименования полей и таблиц, ограничения целостности и т.п.

Для преобразования концептуальной модели в физическую необходимо знать, что:

- Каждая **сущность** в **ER-диаграмме** представляет собой **таблицу** базы данных.
- Каждый **атрибут** становится колонкой (**полем**) соответствующей таблицы.
- В некоторых таблицах необходимо вставить новые атрибуты (поля), которых не было в концептуальной модели — это **ключевые атрибуты родительских таблиц**, перемещённых в **дочерние таблицы** для того, чтобы обеспечить связь между таблицами посредством внешних ключей.

Выводы:

- Семантическое моделирование данных основывается на технологии определения значения данных через их взаимосвязи с другими данными.
- В качестве инструмента семантического моделирования используются различные варианты (нотации) диаграмм сущность-связь — (Entity-Relationship). Нотация — система условных обозначений, принятая в какой-либо области знаний или деятельности.
- ER- диаграммы позволяют использовать наглядные графические обозначения для моделирования сущностей и их взаимосвязей. Основное достоинство метода состоит в том, модель строится методом последовательного уточнения и дополнения первоначальных диаграмм.

Видео-пример выполнения работы

Для выполнения работы рекомендуется использовать бесплатный онлайн-сервис draw.io <https://rt.draw.io/>.

Пример выполнения можно посмотреть по ссылке: <https://youtu.be/uKImrwjOKTU> или перейти с использованием QR-кода.



Работа 6 проектирование диаграммы состояний информационной системы в нотации UML

Создание диаграммы состояний

Цель работы: получить навыки построения диаграмм UML.

Задание: разработать диаграмму состояний для одного из ранее разработанных классов или прецедентов.

Практическая работа № 3

Тема. Разработка и создание UML диаграмм.

Цель. Создание в среде ArgoUML диаграмм UML.

Ход работы

1. Ознакомиться с теоретической частью.
2. Выполнить практическое задание.
3. Ответить на контрольные вопросы.
4. Оформить отчет.

Теоретическая часть

После запуска ArgoUML появится окно. Окно ArgoUML состоит из следующих компонентов: меню, панели инструментов и четырех панелей. Вверху слева находится Explorer, затем идет панель редактирования, затем на нижней панель деталей и To-Do панель.

В панели Explorer содержится список всех классов, интерфейсов и типов данных модели в виде дерева. На панели редактирования можно редактировать диаграммы.

В панели To-Do отображаются элементы моделей.

Описание пунктов меню:

Файл (File) — позволяет создавать новые проекты, сохранять и открывать проекты, импортировать, импортировать источники из другого расположения, скачать и сохранить модель из/в базу данных, печать модель, сохранить модель, сохранять конфигурацию модели и выход из программы.

Редактировать (Edit) -позволяет выбрать один или несколько UML элементов в диаграмме, а также повторить действия, перемещать элементы из диаграммы и так далее

Вид (View) -позволяет переключаться между диаграммами, искать артефакты в модели, масштабировать диаграмму, выберите особенное представление диаграмм и т.п.

Создать диаграмму (Create Diagram) -позволяет создавать любую диаграмму из семи UML диаграмм, поддерживаемых ARGUML (class, use case, state, activity, collaboration, deployment и sequence).

Расставить (Arrange) – позволяет выровнять, распределить, упорядочить артефакты в диаграмме.

Генерация кода (Generation) -позволяет сгенерировать Java код для отдельных или всех классов.

Критика (Critique) –переключать авторецензирование, устанавливать уровень важности проблемы дизайна и дизайн целей и увидеть критических изменения. Инструменты (Tools) Помощь (Help)

Практическая часть

Задание 1. Для создания диаграммы вариантов использования, выполните следующие действия:

- 1) Дважды щелкните на значке untitledModel.
- 2) Нажмите на значок Диаграмма вариантов использования в проводнике, чтобы открыть диаграмму Вариантов использования.
- 3) С помощью кнопки Use Case (вариант использования) на панели инструментов поместите на диаграмму новый вариант использования.
- 4) Назовите этот новый вариант «Ввести новый заказ», для этого вам нужно дважды нажать на варианте использования и ввести имя или активировать вкладку Свойства To-Do панели в поле Имя: введите имя.
- 5) Повторите шаги 3 и 4, чтобы разместить на диаграмме остальные варианты.
- 6) С помощью кнопки Actor (Действующее лицо) инструментов поместите на диаграмму новое действующее лицо
- 7) Назовите его «Продавец», действия аналогичные вариантам использования.
- 8) Повторите шаги 6 и 7, поставив на диаграмме остальных актеров.

Задание 2. Для указания абстрактного варианта использования, выполните следующие действия:

- 1) ЛКМ нажмите на вариант использования «Отклонить заказ» на диаграмме.
- 2) В панели Свойств в области modifiers выберите контрольный переключатель isAbstract (Абстрактный), чтобы сделать этот вариант использования абстрактным.

Задание 3. Для добавления ассоциации выполните следующие действия:

- 1) Нажав на кнопку Association (Ассоциация), выберите UniAssociation (Однонаправленная Ассоциация), изобразите ассоциацию между актером Продавец и вариантом использования «Ввести новый заказ».
- 2) Повторите этот шаг, чтобы разместить на диаграмме остальные ассоциации.

Задание 4. Для добавления связи расширения выполните следующие действия:

Нажав на кнопку Extend на панели инструментов нарисуйте связь между вариантом использования «Отклонить заказы» и вариантом использования «Оформить заказ». Стрелка должна быть протянута от первого варианта использования ко второму. Связь расширения означает, что вариант использования «Отклонить заказ», при необходимости, дополняет функциональные возможности варианта использования «Оформить заказ». Диаграмма должна иметь вид, как на рисунке 1.

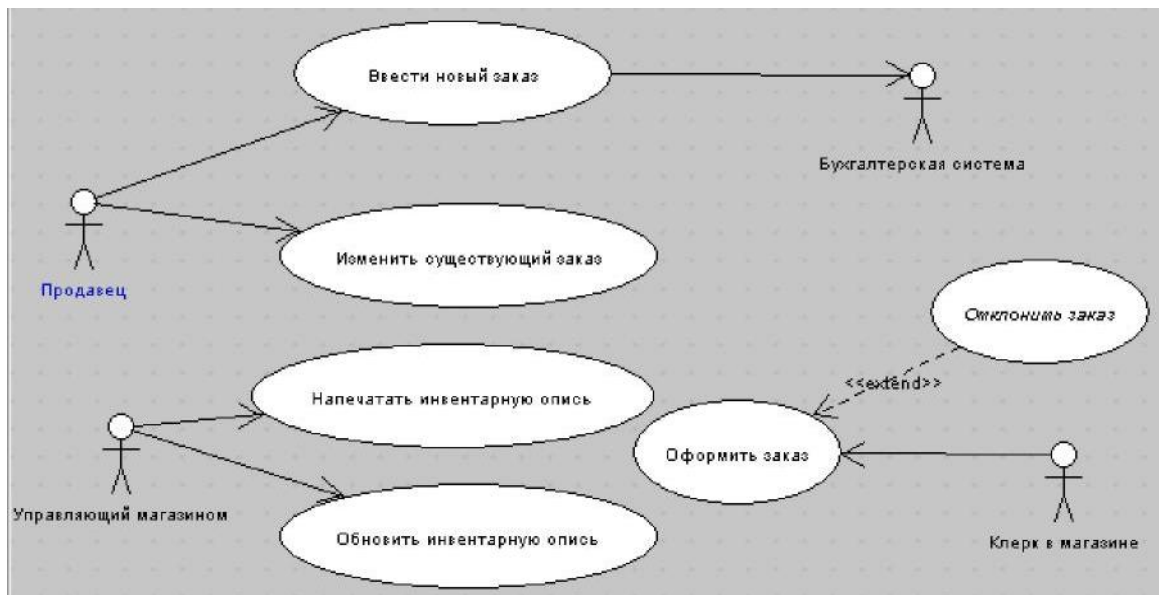


Рисунок 1 – Диаграмма вариантов использования для системы обработки заказов

Задание 5. Создать главную диаграмму классов.

1) Дважды щелкните на значке UntitledModel, отобразится содержимое пакета. Выберите Диаграмма классов. С помощью кнопки Новый пакет и панели инструментов разместите на диаграмме новый пакет. Во вкладке Свойства, укажите имя пакета Entities (Сущность).

2) Создайте пакеты Boundaries (Границы) и Control (Управление).

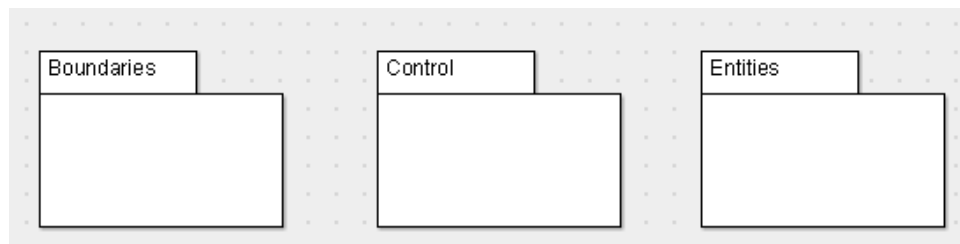


Рисунок 2 – Главная диаграмма классов системы обработки заказов

Задание 6. Создание диаграммы классов для сценария «Ввести новый заказ» со всеми классами

- 1) В меню Создать диаграмму, выберите Диаграмма классов.
- 2) Во вкладке Свойства введите имя для новой диаграммы класса Add New Order (Ввод нового заказа).
- 3) Щелкните в браузере на этой диаграмме, чтобы открыть ее.
- 4) Создайте в окне диаграммы классов классы OrderOptions (Выбор заказа), OrderDetail (Детали заказа), Order (Заказ), OrderMgr (Менеджер заказов) и TransactionMgr (Менеджер транзакций). Имя класса вводится во вкладке Свойства.
- 5) Добавьте операции, указанные на диаграмме классов (рис.3). Чтобы сделать это, выберите соответствующий класс, нажмите на кнопку Добавить операцию, здесь вы можете ввести видимость операции. Диаграмма классов должен выглядеть как на рисунке 3.

Задание 7. Добавление стереотипов в классы

- 1) Нажмите на класс OrderOptions.
- 2) Введите в поле имя класса стереотип <<Boundary>>.

- 3) В раскрывающемся списке стереотипов теперь будет стереотип Boundary . Укажите его.
- 4) Нажмите с помощью мыши на классе OrderDetail.
- 5) Из раскрывающегося списка Стереотип выберите стереотип Boundary.
- 6) Свяжите классы OrderMgr и TransactionMgr со стереотипом Control, а класс Order – со стереотипом Entity. Теперь диаграмма классов должна выглядеть как на рисунке 3.

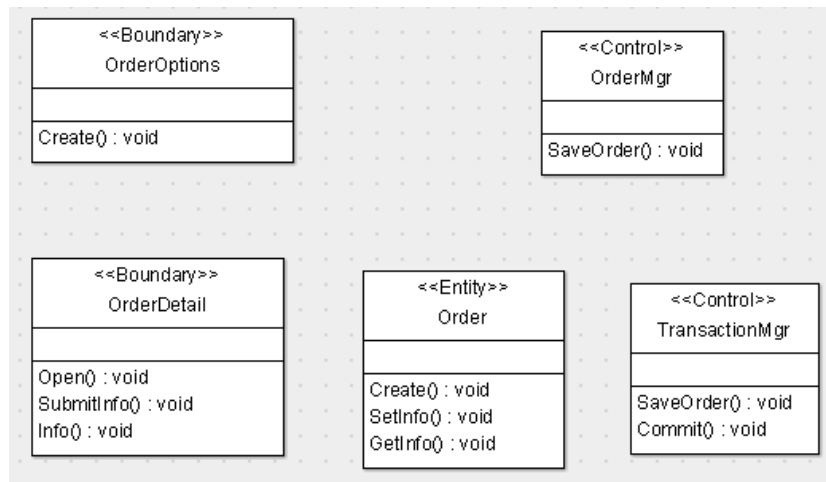


Рисунок 3 – Стереотипы классов для варианта использования Ввести новый заказ

Задание 8. Объединение классов в пакеты

- 1) Перетащите в браузер класс OrderDetail на пакет Boundaries.
- 2) Перетащите класс OrderOptions на пакет Boundaries.
- 3) Перетащить классы OrderMgr и TransactionMgr на пакет Control.
- 4) Перетащите класс Order на пакет Entities.

Задание 9. Добавить диаграмму классов к каждому пакету

- 1) Дважды щелкните левой кнопкой на пакете Boundaries в окне диаграмм.
- 2) В появившемся окне выберите Да.
- 3) Во вкладка Свойства, введите имя для новой диаграммы – MainB.
- 4) Щелкните на диаграмму, чтобы открыть его.
- 5) В браузере выделите класс OrderOptions (потом OrderDetail), щелкните ПКМ и выберите опцию Добавить в диаграмму, а затем нажмите мышью в поле диаграммы.
- 6) Классы будут отображаться на диаграмме.
- 7) Выполните эти же шаги, чтобы поместить классы OrderMgr и TransactionMgr на главную диаграмму классов пакета Control (MainC).
- 8) Выполните эти же действия, чтобы поместить класс Order на главную диаграмму классов пакета Entities (MainE).

Задание 10. Для того чтобы создать диаграммы кооперации выполните следующие действия:


- 1) Нажмите на значок untitledModel в браузере.
- 2) Из меню Создать диаграмму, чтобы выберите Диаграмму коопераций.
- 3) Назовите эту диаграмму Ввод заказа.
- 4) Нажмите на нее, чтобы открыть.

Задание 11. Для добавления объектов к диаграмме коопераций, выполните следующие действия:

- 1) Перетащите с диаграммы вариантов использования объект Продавец на рабочую область.
- 2) На панели инструментов нажмите кнопку ClassifierRole.
- 3) Щелкните в любом месте диаграммы, чтобы разместить там новый объект.

- 4) Назовите объект «Order Options Form».
- 5) Помести на диаграмме остальные объекты: Order Detail Form, Order Manager, Transaction Manager, Order #1234.

Задание 12. Добавления сообщений на диаграмму коопераций:

- 1) На панели инструментов нажмите кнопку Новая ассоциация.
- 2) Свяжите Продавец с объектом Order Options Form.
- 3) Соедините остальные объекты.
- 4) Нажмите на связь между Продавцом и Order Options Form. На панели инструментов нажмите кнопку Добавить сообщение. Нажмите на связь между Продавцом и Order Options Form. Выбрав сообщение, введите его имя Create ().
- 5) Поместите на диаграмме оставшиеся сообщения: Open (), SubmitInfo (), Save (), SaveOrder(), SetInfo (), GetInfo ().
- 6) Нажмите на объекте для добавления к нему сообщения рефлексии. По бокам объекта будут показаны значки, среди которых необходимо выбрать .
- 7) Нажмите на связь рефлексии Transaction Manager, чтобы ввести сообщение. На панели инструментов нажмите кнопку Добавить сообщение. Назовите новое сообщение Commit () (Сохранить информацию о заказе в базе данных) (рисунок 4).

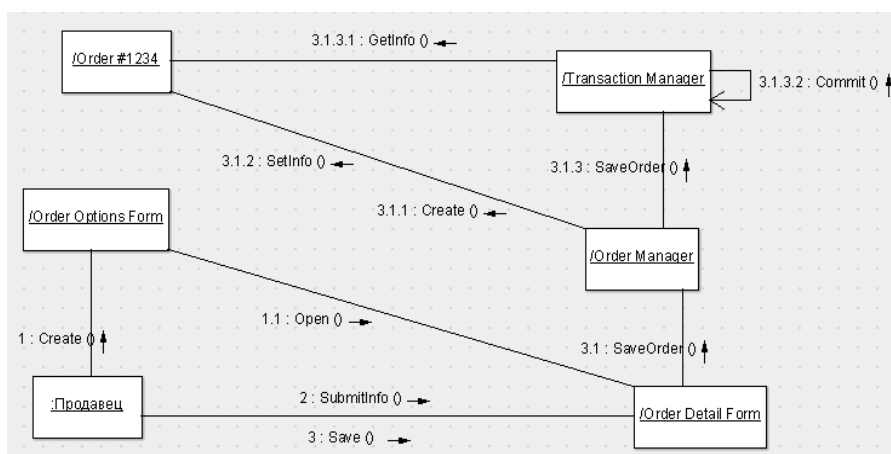


Рисунок 4 – Диаграмма коопераций

Задание 13. Для создания схемы последовательностей, выполните следующие действия: щелкните ПКМ логическое представление браузера. В открывшемся меню нажмите Create Diagram → Диаграмма последовательности. Назовите новую диаграмму «Ввод заказ». Дважды щелкните на нем, чтобы открыть

Задание 14. Для добавления на диаграмму актера и объектов, выполните следующие действия:

- 1) Перетяните актера Продавец из браузера на диаграмму. Для панели панель инструментов, нажмите кнопку New Classifier Role. Имя объекта Order Options Form – Выбор варианта заказа.
- 2) Таким же образом добавьте все элементы, описанные в пункте 11.5 (рисунок 5).

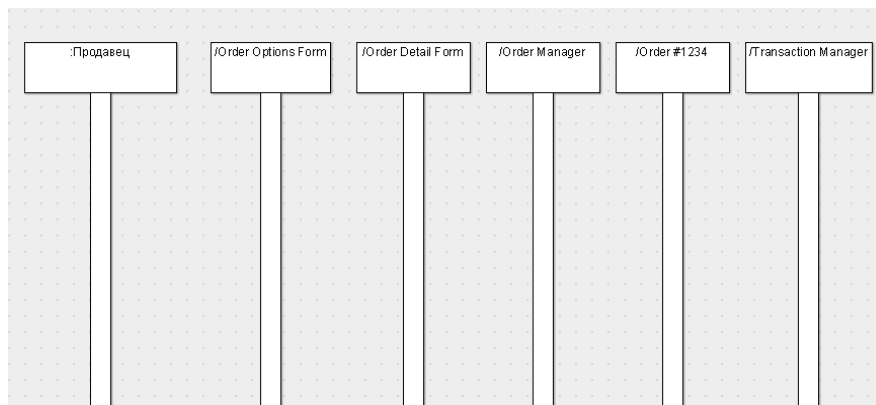


Рисунок 5 – Диаграмма последовательности

Задание 15. Для добавления сообщений на диаграмму, выполните следующие действия:

1) На панели инструментов нажмите кнопку Действие отправки, перетащите от линии жизни актера к линии жизни объекта Order Options Form. В ArgoUML в данном виде диаграмм сообщения автоматически не нумеруются, поэтому нумерация переносится из диаграммы кооперации. Выбрав сообщение, введите имя 1: Create ().

2) Добавьте оставшиеся сообщения, описанные в пункте 12.5

Задание 16. Сопоставление объектов с классами

1) Нажмите на объекте Order Options Form.

2) В панели Свойств раскройте список База.

3) В открывшемся списке нажмите на кнопку с плюсом..

4) В открывшемся окне выберите класс OrderOptions и нажмите на кнопку со стрелкой для добавления в список.

5) Нажмите кнопку ОК.

6) Сопоставьте остальные объекты с классами: Order Detail Form - OrderDetail, Order Manager – OrderMgr, Transaction Manager - TransactionMgr, Order #1234 - Order.

Задание 17. Соотношение сообщений с операциями.

1) Нажмите на сообщение 1: Create () заказ.

2) В окне Свойств дважды щелкните в поле Действие. В открывшемся окне введите Create ().

3) Сопоставьте остальные сообщения.

Задание 18. Добавление нового класса

1) Найдите в дереве (браузере) диаграмму классов «Add New Order». Нажмите на нее дважды, чтобы открыть.

2) Поместите на этой диаграмме новый класс OrderItem (Позиция заказа). Назначьте этому классу стереотип Entity.

3) В дереве перетащите класса в пакет Entities.

Задание 19. Добавление атрибутов

1) Щелкните ПКМ на классе Order (Заказ).

2) В открывшемся меню выберите Добавить → Новый атрибут.

3) Введите новый атрибут OrderNumber : Integer и нажмите Enter. Тип атрибута можно выбрать из раскрывающегося списка в поле Тип или ввести вручную.

4) Введите следующий атрибут CustomerName : String.

5) Добавьте атрибуты OrderDate : Date и OrderFillDate : Date.

6) Щелкните ПКМ на классе OrderItem.

7) В открывшемся меню выберите Добавить → Новый атрибут.

8) Введите новый атрибут ITEMID : Integer.

9) Введите следующий атрибут ItemDescription : String.

Задание 20. Добавления операций к классу OrderItem

- 1) Щелкните ПКМ на классе OrderItem.
- 2) В открывшемся меню выберите Добавить → Новая операция.
- 3) Введите новую операцию Create.
- 4) Введите следующую операцию SetInfo.
- 5) Введите следующую операцию GetInfo.

Задание 21. Подробное описание операций с помощью диаграммы класса

- 1) Нажмите на класс Order, выбрав его таким образом.
- 2) Нажмите на этот класс в разделе операций.
- 3) Отредактируйте операцию Create() так, чтобы она выглядела таким образом:
Create () : Boolean. Для этого дважды нажмите на операцию Create в разделе операций класса.
- 4) Отредактируйте операцию SetInfo(), чтобы она выглядела следующим образом: SetInfo(OrderNum : Integer, Customer : String, OrderDate : Date, FillDate : Date) : Boolean
- 5) Отредактируйте операцию GetInfo (), чтобы она выглядела следующим образом: GetInfo () : String

Задание 22. Подробное описание операций остальных классов

- 1) С помощью браузера или диаграмм, введите следующую сигнатуру операций класса OrderDetail: Open () : Boolean, SubmitInfo() : Boolean, Save() : Boolean.
- 2) С помощью браузера или диаграмм, введите следующую сигнатуру операций класса OrderOptions : Create () : Boolean .
- 3) С помощью браузера или диаграмм, следующую сигнатуру операций класса OrderMgr: SaveOrder(ORDERID: Integer): Boolean.
- 4) С помощью браузера или диаграмм, введите следующую сигнатуру операций класса TransactionMgr: SaveOrder (ORDERID: Integer): Boolean, Commit (): Integer.

Задание 23. Добавление отношения между классами.

Чтобы найти отношения, были изучены диаграммы последовательности. Все классы, которые там взаимодействуют, требовали определения соответствующих связей на диаграммах классов. После выявления связей их добавили в модель. Добавьте отношения, как показано на рисунке 6.

Множественность указывается выбором из контекстного меню связи опции Multiplicity. После добавления связей диаграмма классов должна выглядеть следующим образом.

Задание 24. Создание диаграммы состояний.

- 1) Найдите в браузере класс Order.
- 2) Выделите этот класс.
- 3) В меню Создать диаграмму выберите Диаграмма состояний.

Задание 25. Добавление начального и конечного состояний.

- 1) На панели инструментов нажмите кнопку New Initial (Начальное состояние). Поместите это состояние на диаграмму.
- 2) На панели инструментов нажмите Конечное состояние. Поместите на диаграмму.

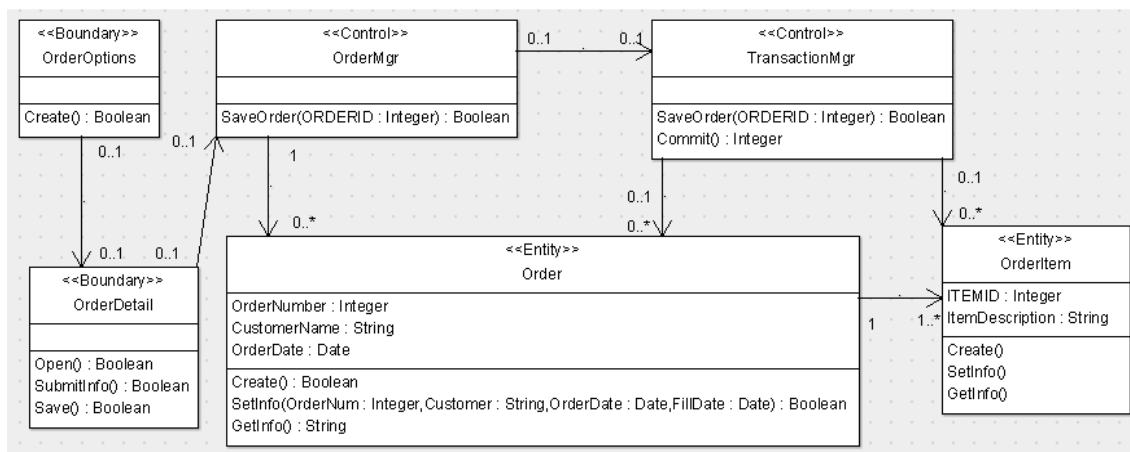


Рисунок 6 – Диаграмма классов Add New Order

Задание 26. Добавление суперсостояния.

- 1) На панели инструментов нажмите кнопку Композитное состояние.
- 2) Поместите это состояние на диаграмму.

Задание 27. Добавление оставшихся состояний.

- 1) На панели инструментов нажмите кнопку Простое состояние. Поместите его на диаграмму. Назовите состояние Отменен.
- 2) На панели инструментов нажмите кнопку Простое состояние. Поместите его на диаграмму. Назовите состояние Выполнен.
- 3) На панели инструментов нажмите кнопку Простое состояние. Поместите это состояние внутрь суперсостояния. Назовите состояние Инициализация.
- 4) На панели инструментов нажмите кнопку Простое состояние. Поместите это состояние внутрь суперсостояния. Назовите состояние Сборка заказа.

Задание 28. Подробное описание состояний

Для добавления в состояния действий на входе, выходе и деятельности исполнения на вкладке Свойства состояния выберите соответственно: действие при входе, действие при выходе, деятельность выполнения.

Задание 29. Добавление переходов

- 1) На панели инструментов нажмите кнопку New Transition (Переход).
- 2) Нажмите на начальной точке. Проведите линию перехода к состоянию Инициализация.
- 3) Повторить предыдущие шаги для добавления оставшихся переходов.

Задание 30. Подробное описание переходов

- 1) Нажмите на переход от состояния Инициализация к состоянию Сборка заказа, открывая окно его свойств.
- 2) В поле Имя введите Выполнить заказ.
- 3) Повторите этапы, добавив событие Отменить заказ для переключения между суперсостоянием и состоянием Отменен.
- 4) Нажмите на переход от состояния Сборка заказа к состоянию Заполнены (Выполнено), открывая окно его свойств.
- 5) В поле Имя введите фразу Добавить Порядок пункта (Добавить в заказ на новое место).

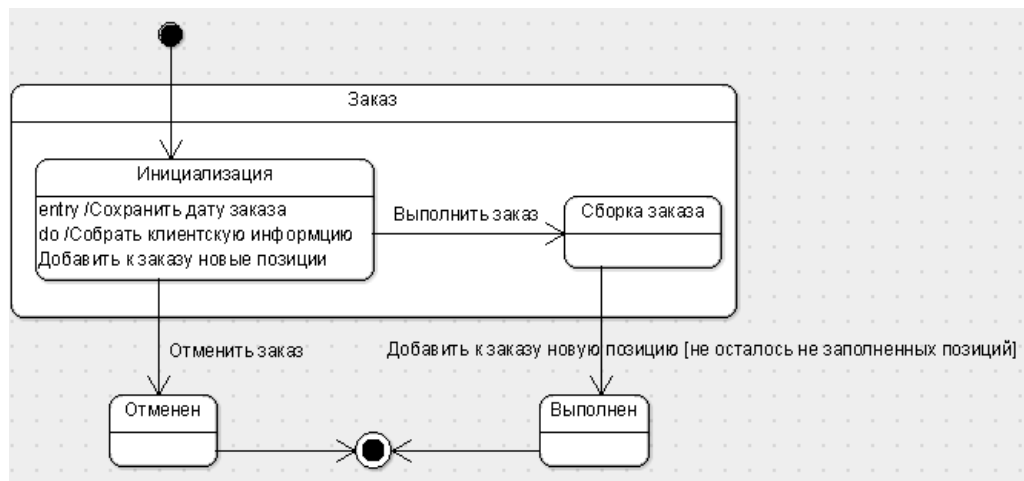


Рисунок 7 – Диаграмма состояний

Задание 31. Добавление компонентов и изображение зависимостей

- 1) Выполните команду Создать диаграмму → Диаграмма развертывания.
- 2) Нажмите на название диаграммы в браузере. Введите имя Entities.
- 3) На панели инструментов нажмите кнопку New Component (Компонент).
- 4) Щелкните в поле диаграммы. Введите имя компонента.
- 5) Добавить все компоненты.
- 6) На панели инструментов нажмите кнопку New Dependency (Зависимость).

Проведите необходимые зависимости. Аналогично создайте диаграммы Control и Boundaries.

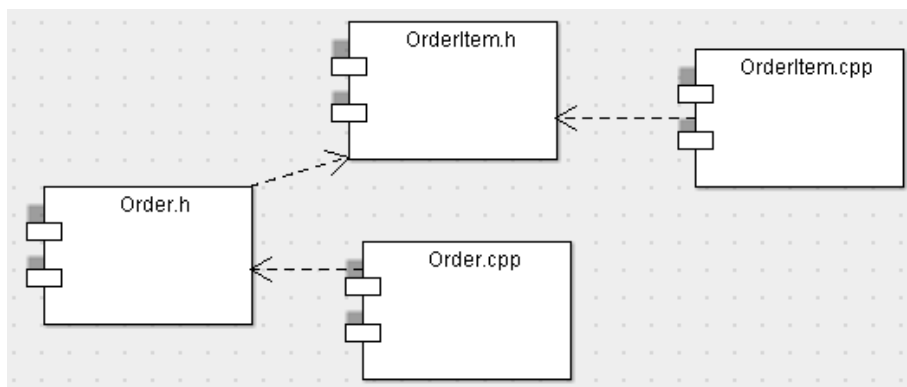


Рисунок 8 – Диаграмма компонентов Entities

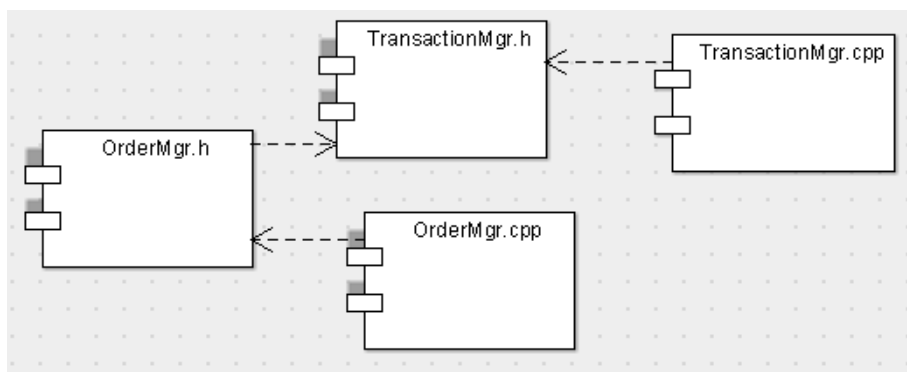


Рисунок 9 – Диаграмма компонентов Control

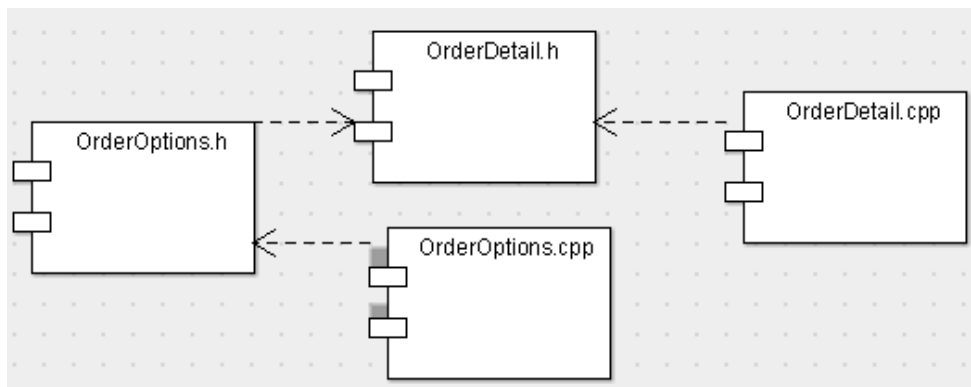


Рисунок 10 – Диаграмма компонентов Boundaries

Задание 32. Создание диаграммы компонентов системы

- 1) Выполните команду Создать диаграмму → Диаграмма развертывания.
- 2) Нажмите на название диаграммы Диаграмма компонентов системы.
- 3) Перетащите из дерева на рабочую область диаграммы компоненты OrderDetail.h, OrderOptions.h, OrderMgr.h, TransactionMgr.h, Order.h, OrderItem.h.
- 4) Создайте дополнительные компоненты OrderClient.exe и OrderServer.exe.
- 5) Создайте зависимости между компонентами, которые не были созданы автоматически.

Задание 33. Добавление узлов на диаграмму размещения

- 1) Выберите пункт меню Создать диаграмму → Диаграмма развертывания.
- 2) На панели инструментов нажмите кнопку New Node.
- 3) Введите имя узла «Сервер базы данных».
- 4) Добавьте оставшиеся узлы.

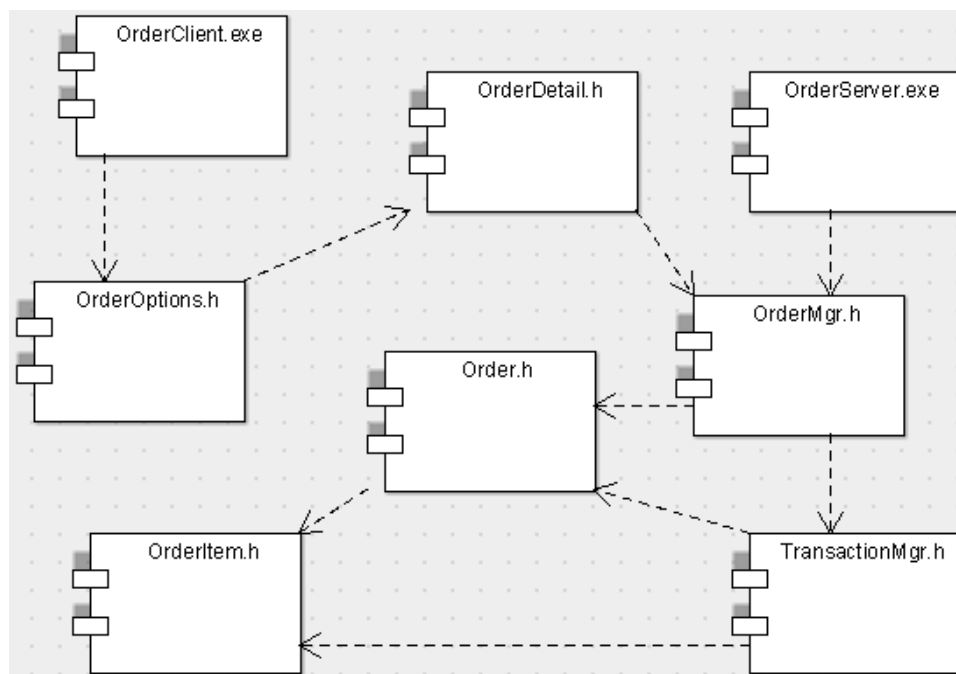


Рисунок 12 – Диаграмма компонентов системы

Задание 34. Добавление связей

- 1) На панели инструментов нажмите кнопку New Link.
- 2) Нажмите на «Сервер базы данных».
- 3) Проведите линию до узла «Сервер приложений».

4) Добавьте остальные связи.

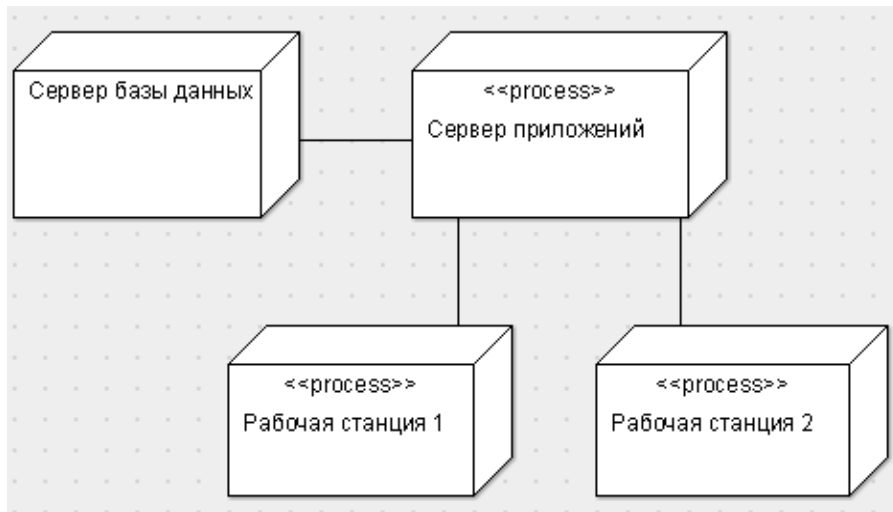


Рисунок 13 – Диаграмма размещений

Задание 35. Добавление процессов и операций

- 1) Щелкните ПКМ на «Сервер приложений». Из раскрывающегося списка выберите Apply Stereotypes → <<process>>.
- 2) Выполните аналогичные действия для объектов Рабочая станция 1 и Рабочая станция 2.
- 3) Выделите «Сервер приложений». На панели Свойств в поле Операции нажмите ПКМ и выберите Новая операция. В поле Имя введите OrderServerExe.
- 4) Добавьте остальные процессы: Рабочая станция 1 – OrderClientExe, Рабочая станция 2 – ATMClientExe.

Задание 36. Результаты выполнения практического задания запишите в отчет.

Контрольные вопросы

1. Для чего используется язык UML?
2. Опишите основные диаграммы UML.

Содержание отчета

1. Тема. Цель. Оборудование.
2. Результат выполнения практического задания.
3. Ответы на контрольные вопросы. Вывод.

Содержание отчета: диаграмма состояний и описание состояний в виде таблицы:

Состояние	Описание состояния

Пример выполнения работы

Диаграммы состояний применяются, как правило, для моделирования поведения классов, прецедентов или системы в целом.

Составим диаграмму состояний для класса *Order (Заказ)*, поскольку в нашей модели он наиболее часто будет менять свое состояние. Заказ может находиться в нескольких состояниях:

- при создании заказа он переходит в состояние *Инициализация*, в котором выполняются некоторые предварительные действия;
- после завершения инициализации заказ переходит в состояние *Открыт*, в котором к заказу добавляются новые пункты. Выход из этого состояния возможен или в случае отмены заказа, или в случае заполнения всех необходимых пунктов заказа;
- если заполнены все необходимые пункты заказа, то он переходит в состояние *Закрыт*, в котором происходит выписка счета. Выход из этого состояния произойдет только после того, как счет будет выписан;
- если заказ отменен, то из состояния *Открыт* он переходит в состояние *Отменен*. При выходе из этого состояния происходит удаление всех пунктов заказа.

Диаграмма состояний для класса *Order* представлена на рисунке 5.1.

Первым состоянием на диаграмме состояний является начальное состояние. При выполнении события "заказ создан" заказ переходит в состояние *Инициализация*. При входе в это состояние выполняется входное действие "Сохранить дату заказа". Основное действие, которое будет выполняться в течении всего времени, пока заказ будет находиться в этом состоянии, это "Внести информацию о клиенте". Переход из этого состояния в состояние *Открыт* произойдет только при выполнении сторожевого условия "инициализация завершена".

В состоянии *Открыт* имеется выходное действие и переход в себя. Переход в себя означает, что событие инициирует переход, происходит выход из текущего состояния, выполняется некоторое действие, после чего происходит возврат в исходное состояние. Поскольку при переходе в себя происходит выход из состояния и повторный вход в него же, то выполняется

действие, ассоциированное с переходом, и, кроме того, действие при входе в состояние. В состоянии *Открыт* к заказу добавляются новые пункты, причем их можно добавить только в том случае, если есть незаполненные пункты. Для показа этого мы использовали переход в себя "Добавление пункта заказа" со сторожевым условием "заполнены не все пункты заказа". Выход из этого состояния состоится в двух случаях - или когда выполнится сторожевое условие "заполнены все позиции заказа" (при этом заказ перейдет в состояние *Закрыт*), или когда наступит событие "заказ отменен" (при этом заказ перейдет в состояние *Отменен*). При выходе из состояния выполнится действие выхода "* OrderItem.Create()" (создание пункта заказа). Символ "*" указывает на то, что это действие выполнится много раз (по числу добавленных пунктов в заказ).

В состоянии *Закрыт* присутствует только внутреннее действие - "Выписать счет". В это состояние заказ переходит из состояния *Открыт* только при выполнении сторожевого условия "заполнены все позиции заказа". Выход из этого состояния и переход в конечное произойдет при наступлении события "счет выписан".

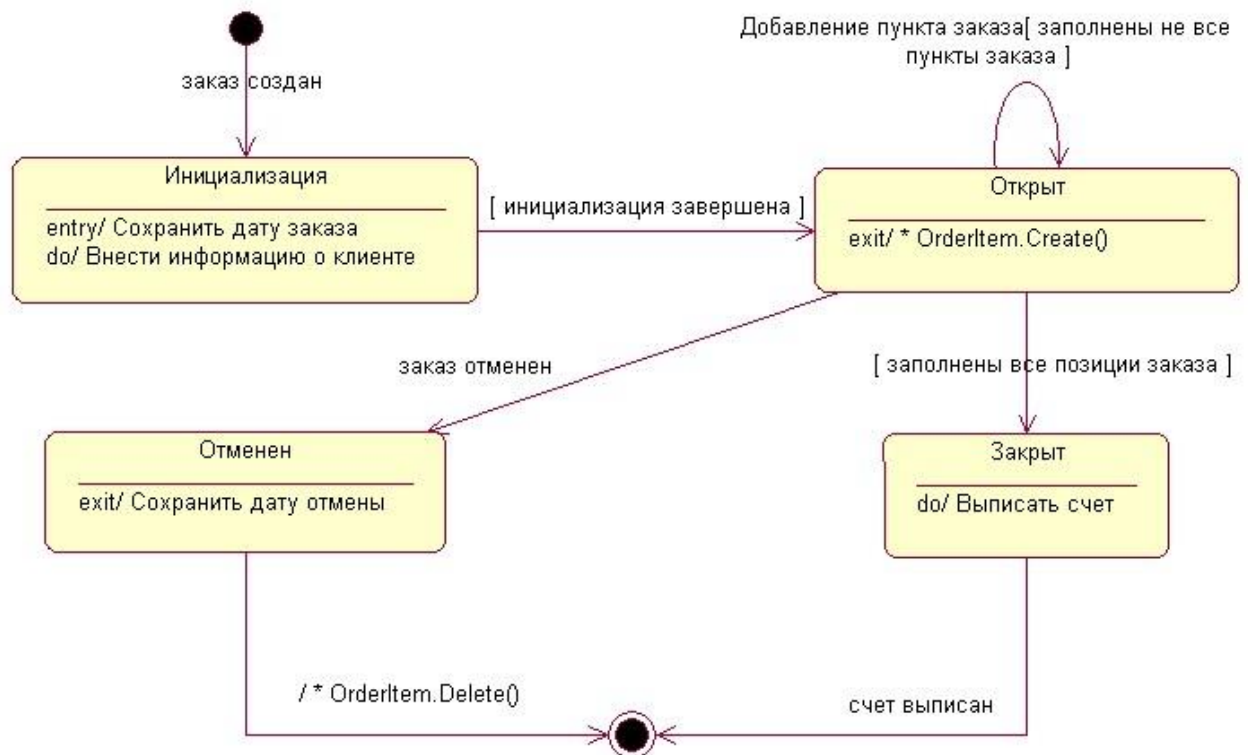


Рисунок 5.1 – Диаграмма состояний для класса Order

В состояние *Отменен* заказ переходит из состояния *Открыт* при наступлении события "заказ отменен". При выходе из него выполняется действие выхода "Сохранить дату отмены". При переходе из этого состояния в конечное выполняется действие "** OrderItem.Delete()*" (удаление пункта заказа). Здесь также стоит "***", поскольку это действие будет выполняться много раз.

Список литературы

1. Петров А. Б. Проектирование информационных систем. Методы анализа для обеспечения безопасности функционирования [Электронный ресурс]: учебное пособие. - М.: РТУ МИРЭА, 2019. - – Режим доступа: <http://media:8080/ebooks/15052019/2007.iso>.
2. Вейцман В. М.. Проектирование информационных систем [Электронный ресурс]: учебное пособие. - Санкт- Петербург: Лань, 2019. - 316 с. – Режим доступа: <https://e.lanbook.com/book/122172>.
3. Рочев К. В.. Информационные технологии. Анализ и проектирование информационных систем [Электронный ресурс]: учебное пособие. - Санкт-Петербург: Лань, 2019. - 128 с. – Режим доступа: <https://e.lanbook.com/book/122181>.

4. Остроух А. В., Суркова Н. Е.. Проектирование информационных систем [Электронный ресурс]: монография. - Санкт-Петербург: Лань, 2019. - 164 с. – Режим доступа: <https://e.lanbook.com/book/118650>.
5. Шелухин О. И., Моделирование информационных систем [Электронный ресурс]:. - Москва: Горячая линия- Телеком, 2018. - 516 с. – Режим доступа: <https://e.lanbook.com/book/111118>.

Работа 7. Создание полного текстового описания, глоссария и расчет параметров проектируемой информационной системы.

Задание: создать полное текстовое описание все процессов и потоков, отображенных в процессе создания и декомпозиции функциональной модели и диаграмм потоков данных проектируемой информационной системы. Вычленив понятия, используемые в полном текстовом описании и создать Глоссарий, дав необходимые определения используемых понятий. Выполнить расчет одного, наиболее важно, параметра информационной системы. Параметр рекомендуется согласовать с преподавателем, ведущим занятия. По умолчанию предлагается выполнить расчет энтропии системы.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Обзор выполнения дан в видео: https://youtu.be/Ds56e6_EVfQ



Описание лабораторного практикума включает в себя учебно-методические материалы к выполнению семи лабораторных работ по всем темам рабочей программы дисциплины «Проектирование информационных систем».

Работа выполняется как во время аудиторных занятий, так и в виде самостоятельной внеаудиторной работы. Выполнение каждой лабораторной работы состоит из трёх этапов:

4. Подготовка и получение допуска к работе.
5. Получение индивидуального задания и выполнение основной части работы (Задание взять из работы 1).
6. Оформление и защита отчёта о проделанной работе.

В начале каждой лабораторной работы выполняется повторение теоретического материала и проверка готовности к выполнению работы с помощью

контрольных вопросов. После получения допуска к выполнению работы выдаётся индивидуальный вариант задания для самостоятельной работы. На заключительном этапе оформляется отчёт о проделанной работе с описанием полученных результатов и выполняется процедура защиты отчёта.

Процедура защиты отчёта заключается в проверке:

- 4) правильности структуры, содержания и оформления отчёта;
- 5) корректности полученных результатов и полноты их описания;
- 6) способности дать объяснение и необходимое обоснование полученным результатам.

Отчет должен включать в себя:

6. Титульный лист.
7. Задание на лабораторную работу.
8. Содержание отчёта.
9. Описание результатов по каждой части задания.
10. Приложение (таблицы данных).

Цели и задачи лабораторной работы

Целями выполнения лабораторной работы являются:

4. Закрепление имеющихся знаний о параметрах ИС. Изучение методологии расчета требуемых параметров проектируемой информационной системы.

5. Приобретение навыков анализа и формализованного описания заданной предметной области.

6. Приобретение навыков расчета параметров информационной системы.

В процессе выполнения лабораторной работы решаются следующие задачи:

4. Выполняется системный анализ заданной предметной области. Составляется формализованное описание информационных объектов предметной области.

5. Выполняется расчет параметров проектируемой информационной системы.

Краткие теоретические сведения

Информация – сообщение, которое имеет ценность, значимость для субъекта. Информация, не обладающая ценностью, называется тривиальной.

Статистически информация – сообщение о состоянии системы, уменьшающее неопределенность знаний о ней. Для измерения

информации вводятся параметры: – количество информации I - объем данных $VД$.

Меры информации	Синтаксическая	Семантическая
Прагматическая	Объем данных $VД$	Ценность использования $Iс$
Количество информации I .		

Синтаксическая мера информации Объем данных $VД$ в сообщении измеряется количеством символов (разрядов) в этом сообщении
Двоичная система счисления: единица измерения – бит (bit – binary digit – двоичный разряд), байт – 8 бит
Десятичная система счисления: единица измерения – дит (десятичный разряд).

Количество информации I на синтаксическом уровне определим с помощью понятия неопределенность состояния системы - энтропия системы - некоторая система $H()$ – мера неосведомленности (неопределенности) о системе $H()$ – неопределенность состояния системы после получения сообщения $I()$ – количество информации о системе, полученной в сообщении (уменьшение неопределенности состояния системы) $I() = H() - H()$.

Энтропия системы может рассматриваться как мера недостающей информации Энтропия системы $H()$, имеющая N возможных состояний по формуле Шеннона P_i –вероятность того, что система находится в i -м состоянии.

$$H(x) = -\sum_{i=1}^n [p_i \cdot \log_a p_i]$$

Семантическая мера измерения информации Для измерения смыслового содержания информации используется понятие тезаурус пользователя.

Контрольные вопросы для допуска к работе

1. Семантическая информация.
2. Энтропия Шеннона.
3. Дисперсия информации.
4. Взаимосвязь данных в реляционных данных, корреляция.
5. Модели данных.
6. Виды и назначение баз данных.
7. Прагматическая информация.
8. Мера информации.

9. Технологии проектирования данных.

10. Анализ параметров проектируемой информационной системы.

Порядок выполнения работы

Порядок выполнения в табличном процессоре Excel можно посмотреть в

видео: https://youtu.be/o4_OT7IHYKE



Вариант индивидуального задания определяет предметную область для разработки проекта базы данных некоторой информационной системы.

В процессе выполнения лабораторной работы необходимо:

7. Составить таблицу данных, оперируемых в проектируемой информационной системе, рассматривая эту таблицу, как диапазон возможных значений данных, хранящихся в будущей информационной системы, автоматизирующей бизнес-процессы некоторой организации.

8. Выполнить анализ заданной предметной области. Сформулировать словесное описание информационных объектов. Описать типовые запросы для поиска и анализа информации об объектах предметной области.

9. Вычислить необходимые промежуточные параметры для вычисления энтропии системы, а также количественные параметры информации в проектируемой ИС.

10. Проверить полноту и корректность выполненных вычислений.

11. Составить результирующую таблицу и выполнить выводы по работе.

Расчет параметров проектируемой информационной системы.

Задание: Выполнить расчет, по крайней мере, одного, наиболее важно, параметра проектируемой информационной системы. Параметр рекомендуется согласовать с преподавателем, ведущим занятия. По умолчанию предлагается выполнить расчет энтропии системы.

ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

4.1 Описание ЭСЕ

Элементарная семантическая единица (ЭСЕ) – неделимая единица информации, используемая в ИС. ЭСЕ представляет собой завершенную контекстную конструкцию, вызываемую в результате поиска по различным атрибутам или в результате тех или иных команд в виде отклика или отчета. В случае исследования настоящей системы за элементарную семантическую единицу была выбрана одна из характеристик поиска, а именно сертификатов возвращаемых на запрос. В нашем примере эта величина меняется случайным образом в пределах от 100000 до 200000 [сертификатов].

4.2. Наполнение системы

Проектируемая информационная система может быть наполнена практически любым количеством элементов базы данных. Их количество ограничиваются только параметрами сервера.

В рамках данной система была наполнена работы Система была наполнена 100 ЭСЕ. В рамках ограничений объема данной лабораторной работы, невозможно привести полный перечень всех записей ЭСЕ, поэтому пример первых десяти записей приведен в таблице 1

Структуризация ведется по количеству сертификатов возвращаемых на запрос.

Таблица 1. Список элементарных семантических единиц.

Наименование	Параметр
Курс	172367
Курс	122059
Курс	124678
Курс	100266
Курс	118235
Курс	112568
Курс	131304
Курс	100612
Курс	140927
Курс	108944

4.3. Математические расчеты

Для дальнейшего исследования проектируемой ИС необходимо рассчитать вероятности, с которыми ЭСЕ принимает то или иное значение. Для оценки этих вероятностей было принято решение разбить весь диапазон значений на 10 дискретных величин с шагом в 10 000. Расчеты ведутся с помощью формулы $P(\xi)=n/N$, где n – благоприятное число исходов (в данном случае число сертификатов, попадающих в данный диапазон), а N – общее число исходов. В таблице 2 приведены возможные значения, принимаемые ЭСЕ и их вероятности.

Таблица 4.2 – Ряд распределения.

№	x	P(x)
1	106215	13/100=0.13
2	115026	11/100=0.11
3	125276	12/100=0.12
4	135447	6/100=0.06
5	144948	6/100=0.60
6	156003	1/100=0.10
7	166492	14/100=0.14
8	177508	6/100=0.60
9	184333	15/100=0.15
10	195108	7/100=0.70

4.4. Расчет математического ожидания информационного блока системы

Математическим ожиданием случайной величины называется сумма произведений всех возможных значений случайной величины на вероятности этих значений. Рассчитаем математическое ожидание для нашей системы, взяв за случайную величину число сертификатов. Расчёт математического ожидания информационного блока на примере 10 записей:

$$Mx_i = \sum_{i=0}^n [p_i \cdot x_i] \quad (4.1)$$

Используя данные, полученные в таблице 2, получаем:

$M(10) = 149185$ [сертификатов], следовательно, наиболее вероятное количество сертификатов на запрос находится в районе 149185 [сертификатов].

4.5. Расчет дисперсии информационного блока системы

$$Dx_i = \sum_{i=0}^n [p_i \cdot (x_i)^2] - \left[\sum_{i=0}^n (p_i \cdot x_i) \right]^2 \quad (4.2)$$

Используя данные, полученные в таблице 2, получаем:

$D(10) = 877025547$ [сертификатов²]

4.6. Расчет среднеквадратического отклонения

$$\sigma_{xi} = \sqrt{Dxi} = \sqrt{877025547}$$

$\sigma_{xi} = 29614,6171$ [сертификатов]

4.7. Расчет энтропии системы

Энтропия системы – это сумма произведений вероятностей различных состояний системы на логарифмы этих вероятностей, взятая с обратным знаком.

$$H(x) = - \sum_{i=1}^n [p_i \cdot \log_a p_i] \quad (4.3)$$

За основание логарифма **a** возьмем двоичную систему счисления.

Энтропия фрагмента информационного наполнения в размере 10 ЭСЕ:

Используя данные, полученные в таблице 2, получаем:

$H(x) = 2,841$ [бит]

4.8. Выводы

В данной главе был осуществлен расчет основных характеристик проектируемой ИС, и получены следующие результаты:

Таблица 4.3 – Параметры проектируемой ИС

математическое ожидание информационного блока	148618,3 [сертификатов]
допустимый разброс значений смысловых информационных блоков (дисперсия)	879242467,3 [сертификатов ²]
СКО	29652,0 [сертификатов]
энтропия информационного наполнения	2,841 [бит]

Список литературы

6. Петров А. Б. Проектирование информационных систем. Методы анализа для обеспечения безопасности функционирования [Электронный ресурс]: учебное пособие. - М.: РТУ МИРЭА, 2019. - – Режим доступа: <http://media:8080/ebooks/15052019/2007.iso>.
7. Вейцман В. М.. Проектирование информационных систем [Электронный ресурс]: учебное пособие. - Санкт-Петербург: Лань, 2019. - 316 с. – Режим доступа: <https://e.lanbook.com/book/122172>.
8. Рочев К. В.. Информационные технологии. Анализ и проектирование информационных систем [Электронный ресурс]: учебное пособие. - Санкт-Петербург: Лань, 2019. - 128 с. – Режим доступа: <https://e.lanbook.com/book/122181>.
9. Остроух А. В., Суркова Н. Е.. Проектирование информационных систем [Электронный ресурс]: монография. - Санкт-Петербург: Лань, 2019. - 164 с. – Режим доступа: <https://e.lanbook.com/book/118650>.
10. Шелухин О. И., Моделирование информационных систем [Электронный ресурс]:. - Москва: Горячая линия- Телеком, 2018. - 516 с. – Режим доступа: <https://e.lanbook.com/book/111118>.

Приложение А

Таблица 3. Полный список элементарных семантических единиц.

Наименование	Параметр
Курс	154248
Курс	177237
Курс	108191
Курс	178588
Курс	190708
Курс	176805
Курс	182143
Курс	163235
Курс	159315
Курс	164788
Курс	126954
Курс	167324
Курс	187887
Курс	109680
Курс	157309
Курс	122670
Курс	121108
Курс	165824
Курс	120036
Курс	142604
Курс	183762
Курс	196406
Курс	166616
Курс	106163
Курс	181640
Курс	110112
Курс	107316
Курс	100789
Курс	156875
Курс	100832
Курс	178187

Купе	115648
Купе	197144
Купе	129737
Купе	144744
Купе	137897
Купе	194455
Купе	164839
Купе	162145
Купе	192112
Купе	113800
Купе	186528
Купе	158214
Купе	112926
Купе	169952
Купе	129916
Купе	198003
Купе	159691
Купе	116859
Купе	130197
Купе	107813
Купе	139147
Купе	107770
Купе	123748
Купе	180942
Купе	124793
Купе	184103
Купе	153477
Купе	185937
Купе	129843
Купе	164534
Купе	178682
Купе	144732
Купе	169492
Купе	114256
Купе	175550

Kype	186544
Kype	108914
Kype	116636
Kype	155880
Kype	196926
Kype	188277
Kype	106803
Kype	168528
Kype	103818
Kype	143314
Kype	145350
Kype	183588
Kype	153414
Kype	164269
Kype	148945
Kype	183466
Kype	120337
Kype	151603
Kype	169982
Kype	114954
Kype	107615
Kype	117310
Kype	183561
Kype	132961
Kype	117462
Kype	105094
Kype	185513
Kype	127476
Kype	134364
Kype	138116
Kype	115327
Kype	181107
Kype	169360
Kype	126695

