Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (ФГБОУ ВО «АмГУ»)

Проектирование АСОИУ

сборник учебно-методических материалов

для направления подготовки 09.03.01 «Информатика и вычислительная техника»

профиль – Автоматизированные системы обработки информации и управления

Благовещенск, 2015

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

Составитель: Бушманов А.В.

Проектирование АСОИУ: сборник учебно-методических материалов для направления подготовки для направления подготовки для направления подготовки 09.03.01 «Информатика и вычислительная техника», профиль – Автоматизированные системы обработки информации и управления.
– Благовещенск: Амурский гос. ун-т, 2015.

Рассмотрен на заседании кафедры «Информационных и управляющих систем».

____._____20____, протокол № ___.

© Амурский государственный университет, 2015

© Кафедра Информационных и управляющих систем, 2015

© Бушманов А.В., составление

1. Краткое изложение лекционного материала

Общая характеристика процесса проектирования АСОИУ

Определение АСОИиУ. Автоматизированная система обработки информации и управления - организованная в единую функциональную целостность совокупность взаимодействующих человеческого персонала, аппаратных, вычислительных, телекоммуникационных средств, программного обеспечения для многоцелевой обработки информации в интересах предоставления результатов людям для принятия решения или же непосредственно для передачи на исполнительные механизмы.

Примеры АСОИиУ и их элементов можно найти во многих сферах деятельности человечества. Это автоматизированные системы управления предприятиями (АСУП), системы автоматизированного проектирования (САПР), банки данных, геоинформационные системы (ГИС), автоматизированные системы научных исследований (АСНИ) и т.д.

Под **проектом** понимается проектно-конструкторская и технологическая документация, в которой представлено описание проектных решений по созданию и эксплуатации ИС в конкретной программно-технической среде.

Объектами проектирования являются отдельные элементы или их комплексы функциональных и обеспечивающих частей. Так, функциональными элементами в соответствии с традиционной декомпозицией выступают задачи, комплексы задач и функции управления. В составе обеспечивающей части ИС объектами проектирования служат элементы и их комплексы информационного, программного и технического обеспечения системы.

В качестве **субъекта проектирования** выступают коллективы специалистов, которые осуществляют проектную деятельность, как правило, в составе специализированной организации, и организация-заказчик, для которой необходимо разработать ИС.

Под **проектированием** понимается процесс преобразования входной информации об объекте проектирования, о методах проектирования и об опыте проектирования объектов аналогичного назначения в проект ИС. С этой точки зрения проектирование ИС сводится к последовательной формализации проектных решений на различных стадиях жизненного цикла ИС: планирования и анализа требований, технического и рабочего проектирования, внедрения и эксплуатации ИС.

Осуществление проектирования ИС предполагает использование определенной технологии проектирования, соответствующей масштабу и особенностям разрабатываемого проекта.

Технология проектирования – это совокупность методологии и средств проектирования ИС, а также методов и средств организации проектирования.

Технология проектирования задается регламентированной последовательностью технологических операций, выполняемых в процессе создания проекта на основе того или иного метода, в результате чего стало бы ясно, не только что должно быть сделано для создания проекта, но и как, кому и в какой последовательности это должно быть сделано.

Знания в области АСОИиУ бурно и динамично развиваются, их объем значителен, а сами АСОИиУ многоплановы, сложны, выдвигают много проблем. Поэтому знать все в области АСОИиУ вряд ли возможно. Но, тем не менее, разумно усвоить:

- фундаментальные, основополагающие принципы;
- прикладные знания о компонентах АСОИиУ;
- способы и методы использования инструментальных средств по АСОИиУ;
- методологию анализа и синтеза систем для проектирования и эксплуатации АСОИиУ.

Смысл специальности “АСОИиУ” - обработка информации, а специалист по “АСОИиУ” - это конструктор и технолог информационных систем различного назначения, обеспечивающий их разработку, внедрение и эксплуатацию.

Центральным звеном АСОИиУ является, безусловно, обработка информации. Поэтому в рамках настоящего курса будут рассмотрены информационные основы управления, затем структура автоматизированных систем (АС) и организация разработки АС.

Структура информационно-логической модели АСОИУ, разработка функциональной модели

Основой процессов управления является переработка информации. Поэтому в разработке автоматизированных систем управления составной частью работ является изучение свойств информации и процессов ее обработки. Совокупность операций по обработке информации и их взаимосвязи на всех этапах ее циркуляции в системе называется *информационным процессом*. Основными операциями информационного процесса являются следующие:

1) сбор информации,
2) представление информации в формализованном виде,
3) занесение информации на носитель, ввод в систему,
4) преобразование (переработка) информации,
5) распределение информации в информационной системе,
6) хранение информации и ее носителей,
7) доступ к информации,
8) выдача информации пользователю.

Таким образом, для создания АСУ важно исследовать суть обработки информации, знать информационные основы управления.

Логическое проектирование включает два этапа:

Первый этап.

1) составление списка всех полей (данных) для описания предметной области и их характеристик, в том числе, наименований, имен, типов, размеров полей;

2) установление логических связей между полями;

3) составление графа связей, образование групп полей, формирование записи, выделение ключей;

4) составление графического и словесного описания логической структуры данных;

5) описание логики взаимосвязи полей и содержания полей;

6) документирование в соответствии с принятыми соглашениями или ГОСТ.

В итоге получаем логическое представление данных.

Завершив первый этап логического проектирования, т.е. создав логическое описание, проводят уточнение структуры данных в зависимости от характера использования информации. На этом этапе логического структурирования выявляют (второй этап, этап уточнения):

1) периодичность и виды обработки информации и ее отдельных частей;

2) виды запросов к информации и возможность их разрешения,

3) вопросы безопасности, целостности, защиты, модифицируемости и другие возможные проблемы обработки информации.

Далее, проводится редактирование логической схемы, ее реструктирование, усовершенствование.

База данных должна адекватно отражать предметную область. Это означает, что должны выполняться следующие условия.

1. Состояние базы данных в каждый момент времени должно соответствовать состоянию предметной области.

2. Изменение состояния предметной области должно приводить к соответствующему изменению состояния базы данных.

3. Ограничения предметной области, отраженные в модели предметной области, должны некоторым образом отражаться и учитываться базе данных.

Практически любая база данных, за исключением совершенно элементарных, содержит некоторое количество программного кода в виде триггеров и хранимых процедур.

В развитых СУБД имеются средства автоматизированного логического проектирования, ведения словарей данных, моделирования запросов и обработки данных. В качестве инструмента проектирования структур данных могут использоваться также системы **CASE** (*Computer Aided System Engineering*) - автоматизированные системы структурного проектирования баз данных и ИС.

Предпроектная стадия, включающая проведение необходимых исследований, работ по формированию структуры АСОИУ и получении в конечном счете технического задания на проектирование может проходить по двум принципиально различным вариантам.

Первый – классический фундаментальный подход. Связан с проведением исследований по вскрытию законов, на основе которых функционирует исследуемая система обработки информации и управления. Далее, вследствие выявленных фундаментальных законов, строится формальная модель, связывающая входы, выходы системы, влияние внешних воздействий на нее. Т.о. получаем формально-математическое представление системы, которое и может в дальнейшем служить основой для функционального, инфологического и техно-рабочего проектирования АСОИУ.

Второй подход, достаточно часто применяемый при построении АСОИУ, включает в себя проведение информационного обследования объекта (предметной области), выявление основных информационных потоков, построения, как правило, имитационной модели функционирования объекта и далее выход также на инфологическое и техно-рабочее проектирование.

Первый подход в большей степени гарантирует получение высококачественной разработки, но требует больших интеллектуальных и финансовых затрат. Второй подход обеспечивает, на первый взгляд, меньшие затраты, но вероятней всего за счет неудачных и малоэффективных решений, общие затраты при реализации работ по второму варианту могут быть и значительно больше, чем в первом случае.

Функциональная модель предназначена для описания существующих бизнес-процессов на предприятии (так называемая модель AS-IS) и идеального положения вещей – того, к чему нужно стремиться (модель TO-BE). Также функциональная модель позволяет идентифицировать все информационные объекты, которыми оперирует предприятие в своей деятельности.

Структурный анализ начинается с исследования того, как организована система управления предприятием, с обследования функциональной и информационной структуры системы управления. По результатам обследования аналитик на первой стадии анализа строит обобщенную логическую модель исходной предметной области, отображающую ее функциональную структуру, особенности основной деятельности и информационное пространство, в котором эта деятельность осуществляется. Аналитик строит модель «как есть».

Вторая стадия работы, к которой привлекаются заинтересованные представители заказчика, а при необходимости и независимые эксперты, состоит в анализе модели «как есть», выявлении ее недостатков и узких мест, определение путей совершенствования системы управления на основе выделенных критериев качества.

Третья стадия анализа, содержащая элементы проектирования, – создание усовершенствованной обобщенной логической модели, отображающей реорганизованную предметную область или ее часть, которая подлежит автоматизации. Эту модель можно назвать моделью «как надо».

На ряду со структурным подходом существует и более мощный подход называемый объектно-ориентированным ООП. Эта методология создана для проектирования больших и сложных систем и имеет ряд преимуществ перед структурным подходом.

ООП базируется на создании объектной модели, которая описывает структуру объектов, составляющих систему, их атрибуты, операции, взаимосвязи с другими объектами.

Цель разработки объектной модели – описать объекты, составляющие в совокупности проектируемую систему, а также выявить и указать различные зависимости между объектами. *Декомпозиция проблемы на объекты* – творческий, плохо формализуемый процесс.

В объектной модели должны быть отражены те понятия и объекты реального мира, которые важны для разрабатываемой системы

Объектная модель имеет четыре главных элемента: абстрагирование, инкапсуляция, модульность, иерархия, без любого из них модель не будет объектно-ориентированной. Кроме главных, имеются еще три дополнительных элемента: типизация, параллелизм, сохраняемость.

Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя. *Инкапсуляция* – это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение; инкапсуляция служит для того, чтобы изолировать обязательства абстракции от их реализации.

Модульность – это свойство системы, которая была разложена на внутренне связные, но слабо связанные между собой модули.

Иерархия – это упорядочение абстракций, расположение их по уровням. *Типизация* – это способ защититься от использования объектов одного класса вместо другого, или по крайней мере управлять таким использованием. *Параллелизм* – это свойство, отличающее активные объекты от пассивных.

Сохраняемость – способность объекта существовать во времени, переживая породивший его процесс, и (или) в пространстве.

Исходные данные для проектирования

Исходные данные обосновываются путем всестороннего рассмотрения условий работы и требований, предъявляемых к АСОИУ, исходя из ее назначения. При этом одной из наиболее важных и трудоемких задач является установление основных показателей системы, которые в наибольшей мере определяют ее качество. Эти показатели часто называют внешними или входными параметрами системы и обозначают $a1$, $a2$, …, an.. К ним относятся, например, среднеквадратическая ошибка воспроизведения сообщений *еэф*, рабочая длина волны l, занимаемая в эфире полоса частот Df, максимальная дальность действия Rmax, среднее время безотказной работы Dtcp, вес G, объем V, время Tp, затрачиваемое на разработку и ввод системы в эксплуатацию, стоимость C и др.

Разработка больших проектов, связанная с работой коллективов размером в несколько десятков и даже сотен человек из нескольких организаций, немыслима без совокупности нормативно-методических документов, регламентирующих различные аспекты процессов деятельности разработчиков. Комплекс таких документов называют нормативно-методическим обеспечением (НМО). Эти документы регламентируют:

• порядок разработки, внедрения и сопровождения ПО;

• общие требования к составу ПО и связям между его компонентами, а также к его качеству;

• виды, состав и содержание проектной и программной документации.

Следование требованиям НМО позволяет создавать ПО высокого качества, соответствующее требованиям международных стандартов в области информационных технологий.

В состав НМО входят стандарты и руководящие документы, методики выполнения сложных операций, шаблоны проектных и программных документов. Все входящие в состав НМО документы классифицируются по следующим признакам:

• виду регламентации (стандарт, руководящий документ, положение, инструкция и т.п.);

• статусу регламентирующего документа (международный, отраслевой, предприятия);

• области действия документа (заказчик, подрядчик, проект);

объекту регламентации или методического обеспечения.

Нормативной базой НМО являются международные и Отечественные стандарты в области информационных технологий и прежде всего:

• международные стандарты ISO/IEC (ISO — International Organization of Standardization - Международная организация по стандартизации, IEC — International Electrotechnical Commission — Международная комиссия по электротехнике);

• стандарты Российской Федерации ГОСТ Р;

• стандарты организации-заказчика.

Процессы создания автоматизированных систем (АС), частью которых является ПО АС, регламентированы стандартами ГОСТ 34.601—90 «Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания»; ГОСТ 34.602-89 «Информационная технология. Комплекс стандартов на автоматизированные системы.

Техническое задание на создание автоматизированной системы» и ГОСТ 34.603—92 «Информационная технология. Виды испытаний автоматизированных систем». Однако процессы создания ПО для современных распределенных систем, функционирующих в неоднородной среде, в этих стандартах отражены недостаточно, а отдельные их положения явно устарели. В результате для каждого серьезного проекта приходится создавать комплекты нормативных и методических документов, регламентирующих процессы, этапы, работы и документы конкретных программных продуктов, поэтому в отечественных разработках целесообразно использовать современные международные стандарты.

ЖЦПО определяется как период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации.

Основным нормативным документом, регламентирующим состав процессов ЖЦ ПО, является международный стандарт ISO/IEC 12207: 1995 «Information Technology - Software Life Cycle Processes». Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО (его российский аналог ГОСТ Р ИСО/МЭК 12207—99 введен в действие в июле 2000 г.).

В данном стандарте процесс определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные. Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными от других процессов, и результатами.

Каждый процесс разделен на набор действий, каждое действие - на набор задач. Каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем не существует заранее определенных последовательностей выполнения (естественно, при сохранении связей по входным данным).

Процесс приобретения (acquisition process) состоит из действий и задач заказчика, приобретающего ПО.

Данный процесс охватывает следующие действия:

1) инициирование приобретения;

2) подготовку заявочных предложений;

3) подготовку и корректировку договора;

4) надзор за деятельностью поставщика;

5) приемку и завершение работ.

Инициирование приобретения включает следующие задачи:

• определение заказчиком своих потребностей в приобретении, разработке или усовершенствовании системы, программных продуктов или услуг;

• анализ требований к системе;

• принятие решения относительно приобретения, разработки или усовершенствования существующего ПО;

• проверку наличия необходимой документации, гарантий, сертификатов, лицензий и поддержки в случае приобретения программного продукта;

подготовку и утверждение плана приобретения, включающего требования к системе, тип договора, ответственность сторон и т.д.

Заявочные предложения должны содержать:

• требования к системе;

• перечень программных продуктов;

• условия и соглашения;

• технические ограничения (например, среда функционирования системы).

Заявочные предложения направляются выбранному поставщику (или нескольким поставщикам в случае проведения тендера). Поставщик — это организация, которая заключает договор с заказчиком на поставку системы, ПО или программной услуги на условиях, оговоренных в договоре.

Подготовка и корректировка договора включают следующие задачи:

• определение заказчиком процедуры выбора поставщика, включающей критерии оценки предложений возможных поставщиков;

• выбор конкретного поставщика на основе анализа предложений;

• подготовку и заключение договора с поставщиком;

• внесение изменений (при необходимости) в договор в процессе его выполнения.

Надзор за деятельностью поставщика осуществляется в соответствии с действиями, предусмотренными в процессах совместной оценки и аудита.

В процессе приемки подготавливаются и выполняются необходимые тесты. Завершение работ по договору осуществляется в случае удовлетворения всех условий приемки.

Процесс поставки (supply process) охватывает действия и задачи, выполняемые поставщиком, который снабжает заказчика программным продуктом или услугой. Данный процесс включает следующие действия:

1) инициирование поставки;

2) подготовку ответа на заявочные предложения;

3) подготовку договора;

4) планирование;

5) выполнение и контроль;

6) проверку и оценку;

7) поставку и завершение работ.

Инициирование поставки заключается в рассмотрении поставщиком заявочных предложений и принятии решения согласиться с выставленными требованиями и условиями или предложить свои.

Планирование включает следующие задачи:

• принятие решения поставщиком относительно выполнения работ своими силами или с привлечением субподрядчика;

• разработку поставщиком плана управления проектом, содержащего организационную структуру проекта, разграничение ответственности, технические требования к среде разработки и ресурсам, управление субподрядчиками и др.

Процесс разработки (development process) предусматривает действия и задачи, выполняемые разработчиком, и охватывает работы по созданию ПО и его компонентов в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала, и т.д.

Процесс разработки включает следующие действия:

1) подготовительную работу;

2) анализ требований к системе;
3) проектирование архитектуры системы;
4) анализ требований к ПО;
5) проектирование архитектуры ПО;
6) детальное проектирование ПО;
7) кодирование и тестирование ПО;
8) интеграцию ПО;
9) квалификационное тестирование ПО;
10) интеграцию системы;
11) квалификационное тестирование системы;
12) установку ПО;
13) приемку ПО.

Подготовительная работа начинается с выбора модели ЖЦ ПО, соответствующей масштабу, значимости и сложности проекта. Действия и задачи процесса разработки должны соответствовать выбранной модели.

Разработчик должен выбрать, адаптировать к условиям проекта и использовать согласованные с заказчиком стандарты, методы и средства разработки, а также составить план выполнения работ.

Анализ требований к системе подразумевает определение ее функциональных возможностей, пользовательских требований, требований к надежности и безопасности, к внешним интерфейсам и т.д. Требования к системе оцениваются исходя из критериев реализуемости и возможности проверки при тестировании.

Проектирование архитектуры системы на высоком уровне заключается в определении компонентов ее оборудования, ПО и операций, выполняемых эксплуатирующим систему персоналом. Архитектура системы должна соответствовать требованиям, предъявляемым к системе, а также принятым проектным стандартам и методам.

Анализ требований к ПО предполагает определение следующих характеристик для каждого компонента ПО:

• функциональных возможностей, включая характеристики производительности и среды функционирования компонента;
• внешних интерфейсов;
• спецификаций надежности и безопасности;
• эргономических требований;
• требований к используемым данным;
• требований к установке и приемке;
• требований к пользовательской документации;
• требований к эксплуатации и сопровождению.

Требования к ПО оцениваются исходя из критериев соответствия требованиям к системе, реализуемости и возможности проверки при тестировании.

Проектирование архитектуры ПО включает следующие задачи (для каждого компонента ПО):

• трансформацию требований к ПО в архитектуру, определяющую на высоком уровне структуру ПО и состав его компонентов;
• разработку и документирование программных интерфейсов ПО и баз данных;
• разработку предварительной версии пользовательской документации;
• разработку и документирование предварительных требований к тестам и плана интеграции ПО.

Архитектура компонентов ПО должна соответствовать требованиям, предъявляемым к ним, а также принятым проектным стандартам и методам.

Разработка модели и защита данных

Большое внимание в настоящее время уделяется вопросам формирования принципов построения механизмов защиты информации (ЗИ) и системы требований к ним. На основе имеющегося опыта можно сформулировать следующие фундаментальные принципы организации защиты информации:

• системность;

• специализированность;

• неформальность.

Основные требования принципа системности сводятся к тому, что для обеспечения надежной защиты информации в современных АСОИУ должна быть обеспечена надежная и согласованная защита во всех структурных элементах, на всех технологических участках автоматизированной обработки информации и во все время функционирования АСОИУ. Специализированность как принцип организации защиты предполагает два аспекта:

1) ввиду специфических особенностей рассматриваемой проблемы надежный механизм защиты может быть спроектирован и организован лишь профессиональными специалистами по защите информации,

2) для обеспечения эффективного функционирования механизма защиты в составе АСОИУ должны функционировать специалисты по защите информации.

В соответствии с данным принципом значительное распространение за рубежом получает специализация по различным аспектам ЗИ. В США, например, на вопросах ЗИ специализируется свыше 100 фирм.

Принцип неформальности означает, что методология проектирования механизма защиты и обеспечения его функционирования в основе своей является неформальной. Эта неформальность интерпретируется в том смысле, что в настоящее время не существует инженерной методики проектирования механизма защиты в традиционном понимании этого термина.

Общие требования к механизму защиты следующие:

1) адекватность, т.е. обеспечение требуемого уровня защиты (определяется степенью секретности подлежащей обработке информации) при минимальных издержках на создание механизма защиты и обеспечение его функционирования;

2) удобство для пользователей, основу чего составляет требование, чтобы механизм защиты не создавал для пользователей дополнительных трудностей, требующих значительных усилий для их преодоления; минимизация привилегий в доступе, предоставляемых пользователям, т.е. каждому пользователю должны предоставляться только действительно необходимые ему права по обращению к ресурсам системы и данным;

3) полнота контроля, т.е. обязательный контроль всех обращений к защищаемым данным; наказуемость нарушений, причем наиболее распространенной мерой наказания является отказ в доступе к системе;

4) экономичность механизма, т.е. обеспечение минимальности расходов на создание и эксплуатацию механизма;

5) несекретность проектирования, т.е. механизм защиты должен функционировать достаточно эффективно даже в том случае, если его структура и содержание известны злоумышленнику.

В автоматизированных банках данных должно быть предусмотрено наличие в них средств идентификации пользователей и ресурсов системы с периодической сменой идентифицирующей информации, многоаспектного разграничения доступа к элементам баз данных (по элементам, по разрешенным процедурам, по условиям операций и др.), криптографического закрытия данных, регистрации обращений к защищаемым данным, контроля за использованием защищаемых данных и т.д.

Основные положения по разработке систем ЗИ могут быть сформулированы так:

1) защита информации является не разовым мероприятием и даже не совокупностью мероприятий, а непрерывным процессом, который должен протекать (осуществляться) во все время и на всех этапах жизненного цикла АСОИУ;

2) осуществление непрерывного процесса защиты информации возможно лишь на базе промышленного производства средств защиты;

3) создание эффективных механизмов защиты может быть осуществлено высококвалифицированными специалистами-профессионалами в области защиты информации;

4) поддержание и обеспечение надежного функционирования механизмов защиты информации в АСОИУ сопряжено с решением специфических задач и поэтому может осуществляться лишь профессионально подготовленными специалистами.

Сохранность информации может быть нарушена в двух основных случаях: при получении несанкционированного доступа к информации и нарушении функционирования ЭВМ. Система защиты от этих угроз включает следующие основные элементы: защиту АСОИУ и ее аппаратуры, организационные мероприятия по обеспечению сохранности информации, защиту операционной системы, файлов, терминалов и каналов связи. Следует при этом иметь в виду, что все типы защиты взаимосвязаны и при выполнении своих функций хотя бы одной из них сводит на нет усилия других. Предлагаемые и реализованные схемы защиты информации в СОД очень разнообразны, что вызвано в основном выбором наиболее удобного и легко осуществимого метода контроля доступа, т.е. изменением функциональных свойств системы.

В качестве классификационного признака для схем защиты можно выбрать их функциональные свойства. На основе и этого признака выделяются системы: без схем защиты; с полной защитой; с единой схемой защиты.

Разработка пользовательского интерфейса

Проектирования интерфейса пользователя – сложная многофакторная и многовариантная задача, требующая системного подхода. В настоящее время считается доказанным, что решение данной задачи заключается не в том, чтобы рационально «вписать» человека в контур управления, а в том, чтобы, исходя из задач управления объектом, разработать систему взаимодействия двух равноправных партнеров (человек-оператор и аппаратно-программный комплекс АСОИУ).

Цель создания эффективного эргономичного пользовательского интерфейса состоит в том, чтобы отобразить информацию настолько эффективно насколько это возможно для человеческого восприятия и структурировать отображение на дисплее таким образом, чтобы привлечь внимание к наиболее важным единицам информации.

Пользователь должен иметь возможность манипулировать объектами в среде приложения. Неплохо, если они (графические элементы) будут ему понятны и станут нести в себе информацию о том, что это такое и что произойдет, если выбрать или произвести действие над каким-то объектом. Необходимо, чтобы пользователь имел наглядное средство отображения информации на различных этапах решения задач, он должен видеть, как его действия влияют на объекты, расположенные на экране.

Создание эффективного интерфейса заключается в быстром, насколько это возможно, развитии у пользователей простой концептуальной модели интерфейса. Концепция согласованности состоит в том, что при работе с компьютером у пользователя формируется система ожидания одинаковых реакций на одинаковые действия, что постоянно подкрепляет пользовательскую модель интерфейса.

Приложение должно проектироваться таким образом, чтобы пользователь в любой момент и на любом этапе работы мог получить помощь, контекстную справку или подсказку.

Безусловно, пользователю нужно дать возможность экспериментировать в приложении (нажатие любых кнопок, изменение настроек и т. д.). Но при этом необходимо из-

бавить его от тупиковых ситуаций: все последствия экспериментов должны быть исправимы, а в лучшем случае еще и обратимы.

Интерфейс должен предоставлять информацию о том, что происходит в данный момент на компьютере. Нельзя допускать, чтобы пользователь долгое время ожидал реакции приложения на некоторое свое действие.

Цветовая гамма, компоновка элементов, пиктограммы, звуки, анимация – все должно помогать пользователю при выполнении задачи. Но здесь важно не переборщить, т.к. в этом случае внимание человека начнет рассеиваться, у него появится раздражение и, как следствие снизится эффективность работы.

Начальным этапом разработки пользовательского интерфейса являются создание его ассоциативной модели, после чего осуществляется проработка концептуального дизайна. Здесь необходимо разработать необходимый набор интерфейсных элементов, каждый из которых должен обладать определенным цветом, формой, надписью и т. п., и все вместе они должны составлять единую систему, вызывающую стойкую систему ассоциаций у пользователей.

Цвет – мощный визуальный инструмент, который необходимо использовать очень осторожно, чтобы не вызвать неадекватной реакции пользователя.

Целесообразно ограничить число цветов до 4 на экране и до 7 для последовательности экранов. Для неактивных элементов рекомендуется использовать бледные цвета. Необходимо использовать цвета согласно представлениям пользователя. Например, для картографа зеленый цвет – лес, желтый – пустыня, синий – вода. Если цвет используется для кодировки информации, необходимо удостовериться, что код адекватно воспринимается пользователем: красный – опасность/стоп, зеленый – нормально/продолжение работы, желтый – предостережение. Для привлечения внимания наиболее эффективны белый, желтый и красный цвета.

Меню необходимый элемент любой автоматизированной системы, позволяющий пользователю выполнять задачи внутри приложения и управлять процессом решения. Достоинство меню в том, что пользователи не должны помнить название элемента или действия, которое они хотят выполнить – они должны только распознать его среди пунктов меню.

Сообщения необходимы для направления пользователя в нужную сторону, подсказок и предупреждений для выполнения необходимых действий на пути решения задачи. Они также включают подтверждения действий со стороны пользователя и подтверждения, что задачи были выполнены системой успешно либо по каким-то причинам не выполнены. Сообщения могут быть обеспечены в форме диалога, экранных заставок и т.п.

Разработка проекта распределенной обработки

Распределенная обработка данных – методика выполнения прикладных программ группой систем.

Сущность ее заключается в том, что пользователь получает возможность работать с сетевыми службами и прикладными процессами, расположенными в нескольких взаимосвязанных абонентских системах. При этом возможны несколько видов работ, которые он может выполнять: удаленный запрос, например, команда, позволяющая посылать одиночную заявку на выполнение обработки данных; удаленная трансакция, осуществляющая направление группы запросов прикладному процессу; распределенная трансакция, дающая возможность использования нескольких серверов и прикладных процессов, выполняемых в группе абонентских систем.

Для распределенной обработки осуществляется сегментация прикладных программ. Передача данных происходит при помощи удаленного вызова процедур либо электронной почты. Первая технология характеризуется высоким быстродействием, а вторая – низкой стоимостью. Известны также программные средства Системы Управления Распре-

деленной Базой Данных (СУРБД), содержатся инструментальные средства распределенной среды обработки данных.

Распределенная среда обработки данных – представляет собой технологию распределенной обработки данных.

Эта среда обычно - набор сетевых служб, предназначенный для выполнения прикладных процессов, рассредоточенных по группе абонентских систем гетерогенной сети.

Системы, имеющие программы распределенной среды, соответственно, являются серверами и клиентами. Серверы связаны (рис. 1) друг с другом логическими каналами, по которым передают друг другу файлы. Каждый сервер имеет свою группу клиентов.

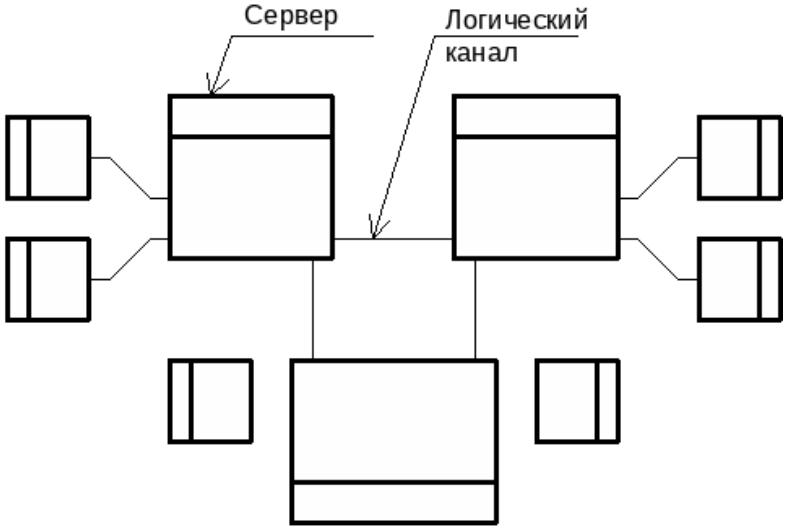


Рис. 1

Среда чаще всего имеет трехступенчатую архитектуру: прикладная программа – база данных – клиент. Функции, выполняемые средой, включают прикладные службы:

- каталогов, позволяющую клиентам находить нужные им серверы;
- интерфейса многопоточной обработки;
- удаленного вызова процедур;
- обслуживания файлов;
- безопасности данных;
- времени, синхронизирующей часы в абонентских системах.

Программное Обеспечение среды погружается, как правило, в Сетевую Операционную Систему. Серверы могут иметь свои, различные операционные системы. В роли сервера может, также, выступать главный компьютер со своей операционной системой.

Функционирование распределенной среды требует выполнения ряда административных задач. К ним, в первую очередь, относятся средства:

- регистрации и контроля за лицензиями пользователей на работу с прикладными программами;
- унифицированных интерфейсов прикладных программ;
- обеспечения безопасности данных;
- инвентаризации программного и технического обеспечения абонентских систем, работающих в сети.

С точки зрения логического управления среда обработки данных делится на ячейки распределенной среды обработки. В каждую из них может включаться от нескольких единиц до тысяч абонентских систем. Размеры ячеек территориально не ограничены. Входящие в одну и ту же ячейку системы могут быть расположены даже на разных континентах.

Структура программных модулей

Программный модуль - это любой фрагмент описания процесса, оформляемый как самостоятельный продукт, пригодный для использования в описаниях процесса.

Каждый программный модуль разрабатывается отдельно от других модулей программы. Каждый разработанный программный модуль может включаться в состав разных программ, если выполнены условия его использования, прописаны в документации этого модуля.

Структура программного обеспечения – совокупность программных компонент, образующих единый программный комплекс в составе АСОИУ. Среди компонент можно выделить приобретаемые и разрабатываемые группы программного обеспечения. Структура разрабатываемого ПО формируется в соответствии с той технологией проектирования ПО, которая выбрана разработчиком АСОИУ.

Сложные задачи целесообразным образом разлагаются на более мелкие подзадачи (“divide-and-concuer” - “разделяй и властвуй”). При этом выбор разложения на подзадачи - структурирование постановки задачи - является одним из важнейших шагов проектирования АСОИУ. Каждая подсистема имеет свою сферу управления с самостоятельными входами и выходами. Результаты решения задач одной подсистемы служат исходными данными или ограничениями для выполнения функций другой подсистемой.

В многоуровневых системах предусмотрены как вертикальные информационные связи, так и горизонтальные. По взаимодействию различают моноиерархические и полииерархические многоуровневые системы. В первых реализуются только радиальные линии передачи информации.

Система программного обеспечения включает в себя:

1. Операционная система – она обеспечивает эксплуатацию ЭВМ однопроцессорных и много процессорных вычислительных комплексов или установок с большим объёмом оперативной памяти и полным набором внешних устройств в разнообразных режимах использования вычислительных систем.
2. Комплексы программ технического обслуживания – обеспечивают проверку правильности функционирования устройств и блоков.
3. Дополнительные тесты – обеспечивают классификацию отказов и локализацию места неисправностей.
4. Пакеты прикладных программ – расширяют возможности ОС.
5. Специальные программы пользователя – основной предмет разработки АСУ, комплекс выполняет свои функции с помощью этих программ.

Разработка программного обеспечения включает в себя следующие этапы:

1. Анализ требований к системе и разработка спецификаций.
2. Разработка алгоритмов и блок-схем программ.
3. Определение структуры программы, выделение программных модулей и их автоматизация.
4. Программирование.
5. Отладка.

Общее программное обеспечение (ОПО) предназначено для организации процесса функционирования вычислительных средств. Оно приобретается или поставляется совместно с ними. К ОПО относятся системы программирования, операционные системы, СУБД, трансляторы языков программирования, служебные и стандартные программы. В качестве ОПО все чаще используется свободное программное обеспечения, в силу его доступности, открытости и соответствия индустриальным стандартам.

Специальное программное обеспечение (СПО) — совокупность программ, реализующих функции (сложные, составные или простые; управляющие, информационные и вспомогательные и т.д.), которые обеспечивают достижения заданных целей создания системы.

В качестве модульной структуры программы принято использовать древовидную структуру, в узлах которой размещаются модули программные, а направленные дуги (стрелки) показывают статическую подчиненность модулей. Дуги показывают, что в тексте модуля, из которого она исходит, есть ссылка на модуль в который она входит. Модульная структура программы, в конечном счете, должна включать в себя всю совокупность спецификаций модулей, из которых она состоит.

Метод восходящей разработки заключается в следующем: Сначала строится модульная структура в виде дерева, затем поочередно программируется модули программы, начиная с самого нижнего уровня. В таком порядке что бы для каждого разрабатываемого модуля был разработан модуль, к которому он может обращаться.

Метод нисходящей разработки заключается в следующем: сначала строится модульная структура в виде дерева, затем поочередно разрабатываются модули программы, начиная с модуля самого верхнего уровня (головного), переходя к разработке какого-либо др. модуля, если уже разработан модуль, который к нему сможет обращаться.

Конструктивный подход – это модификация исходящей разработки, при которой модульная структура программы формируется в процессе разработки модуля. Сначала разрабатывается головной модуль, исходя из спецификации программы в целом, причем спецификация программы одновременно является его головного модуля. В процессе разработки головного модуля, в случае если программа очень большая, выделяется подзадачи или внутренние функции для каждых, из которых создается своя спецификация

Разработка алгоритмов

При разработке алгоритмов учитываются требования пользователей по обеспечению максимального удобства и простоты работы с программой при вводе информации, контролю правильности вводимых параметров, понятности правил и технологии ввода, возможности хранения и отображения графической информации и другие требования.

Алгоритм авторизации

Алгоритм авторизации проверяет совпадение введенной пользователем информации с информацией хранящейся в БД. При неверной авторизации вход не осуществляется.

Алгоритм ввода информации

Информация вводится в соответствующие поля экранной формы и затем сохраняется в базе данных. Благодаря такому способу хранения информации, система может быть перенесена на другую программную платформу.

Алгоритм поиска

При поиске система получает параметры поиска от пользователя, который вводит данные в соответствующие поля. Потом параметры обрабатываются, создается поисковая строка - она добавляется в sql запрос и из базы выбираются записи, которые удовлетворяют критериям поиска.

Алгоритм проверки достоверности реквизитов пластиковой карты

Данный алгоритм включает в себя проверку правильности введенной работодателем информации о сроках действия карты и ее номере. Также проверяется соответствие номера карточки ее типу (VISA, MASTER CARD или AMEX).

Алгоритм оплаты «членства» в системе

Данный алгоритм основан на использовании API интерфейса - интерфейса прикладного программирования. API - это программный интерфейс приложения - набор функций для обработки платежей через Интернет с высокой степенью безопасности.

Преимущества API

Проведение операций с пластиковыми картами через Интернет в защищенной среде - очень сложный процесс, а API упрощает его.

Использование API обеспечивает взаимодействие онлайн оплаты «членства» в системе с центром платежей посредством SSL-протокола - протокола безопасных соедине-

ний. Все операции по пластиковым картам проводятся и управляются через центр платежей.

Рынок интернет-платежей развивается с высокой скоростью и, как следствие, реализующие его технологии и оборудование быстро устаревают. Однако API содержит все необходимые компоненты для безусловного проведения транзакций.

Небольшие системы могут быстро и легко начать использование API для осуществления интернет-платежей. Ознакомившись с порядком использования платежной системы (в данном случае была использована система YourPay) и зарегистрировавшись в ней, первые транзакции можно будет осуществить уже в течение 24 часов. При этом нет необходимости вкладывать дополнительные средства в персонал, оборудование, программное обеспечение, требующиеся для безопасного хранения информации.

SSL - протокол

API предоставляет средства шифрования данных, серверной аутентификации, проверки целостности сообщений и необязательной пользовательской аутентификации для TCP/IP соединения с помощью протокола безопасных соединений (SSL-протокола).

SSL - это протокол, разработанный Netscape для обеспечения безопасной передачи конфиденциальной информации в сети Интернет. Этот протокол использует пары открытых и закрытых ключей для шифрования данных. Открытый и закрытый ключи - различны и каждая пара уникальна. Таким образом, посланные Вами ключи могут Вас идентифицировать.

Открытый ключ предоставляется системе в форме цифрового сертификата, который содержит информацию, способную идентифицировать систему и подтвердить правильность самого ключа. Закрытый ключ - не разглашается и хранится на сервере. Информация, зашифрованная с помощью закрытого ключа, может быть расшифрована только открытым ключом. И наоборот. Этот алгоритм препятствует раскрытию информации в процессе передачи.

Как функционирует API

С помощью модулей API можно сформировать свою собственную платежную систему под конкретные цели. Существуют следующие модули:

- платежный;
- стоимости доставки;
- величины пошлины;
- электронной проверки;
- возврата платежей.

Для проведения операций с пластиковыми картами достаточно использовать только платежный модуль. После интеграции платежного модуля API в систему, работодатель получает возможность оплачивать «членство» через Интернет автоматически, а все детали платежей остаются скрытыми.

Логический анализ структур АСОИУ

Современные технологии разработки программного обеспечения информационных систем предполагают большое разнообразие концепций, подходов и методик. Наиболее популярны из них – это структурный и объектный подходы.

Структурный метод базируется на использовании технологии моделирования потоков данных.

Диаграммы потоков данных (DFD) являются основным средством моделирования функциональных требований к проектируемой информационной системе. С их помощью эти требования представляются в виде иерархии функциональных компонентов (процессов), связанных , потоками данных. Главная цель такого представления – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

В основе методологии лежит построение модели анализируемой системы – проектируемой или реально существующей. В соответствии с методологией модель системы определяется как иерархия диаграмм потоков данных, описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы системы с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. Такая декомпозиция продолжается, создавая многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процесс становятся элементарными и детализировать их далее невозможно.

Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям – потребителям информации. Таким образом, основными компонентами диаграмм потоков данных являются:

· внешние сущности;
· системы/подсистемы;
· процессы;
· накопители данных;
· потоки данных.

Внешняя сущность представляет собой материальный предмет или физическое лицо, представляющее собой источник или приемник информации, например, заказчики, персонал, поставщики, клиенты. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой системы. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь диаграммы анализируемой системы, если это необходимо, или, наоборот, часть процессов может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т.д.

Номер процесса служит для его идентификации. В поле имени вводится наименование процесса в виде предложения с активным недвусмысленным глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже (Ввести сведения о клиентах).

Информация в поле физической реализации показывает, какое подразделение организации, программа или аппаратное устройство выполняет данный процесс.

Накопитель данных представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

Накопитель данных в общем случае является прообразом будущей базы данных и описание хранящихся в нем данных должно быть увязано с информационной моделью.

Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление потока. Каждый поток данных имеет имя, отражающее его содержание.

Первым шагом при построении иерархии ДПД является построение контекстных диаграмм. Обычно при проектировании относительно простых систем строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится так называемый главный процесс, соединенный с приемниками и источниками ин-

формации, посредством которых с системой взаимодействуют пользователи и другие внешние системы.

Если же для сложной системы ограничиться единственной контекстной диаграммой, то она будет содержать слишком большое количество источников и приемников информации, которые трудно расположить на листе бумаги нормального формата, и кроме того, единственный главный процесс не раскрывает структуры распределенной системы.

Признаками сложности (в смысле контекста) могут быть:

· наличие большого количества внешних сущностей (десять и более);

· распределенная природа системы;

· многофункциональность системы с уже сложившейся или выявленной группировкой функций в отдельные подсистемы.

Для сложных систем строится иерархия контекстных диаграмм. При этом контекстная диаграмма верхнего уровня содержит не единственный главный процесс, а набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем.

Разработка контекстных диаграмм решает проблему строгого определения функциональной структуры системы на самой ранней стадии ее проектирования, что особенно важно для сложных многофункциональных систем, в разработке которых участвуют разные организации и коллективы разработчиков.

Для каждой подсистемы, присутствующей на контекстных диаграммах, выполняется ее детализация при помощи ДПД. Каждый процесс на ДПД, в свою очередь, может быть детализирован при помощи ДПД или миниспецификации.

Миниспецификация (описание логики процесса) должна формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу.

При построении иерархии ДПД переходить к детализации процессов следует только после определения содержания всех потоков и накопителей данных, которое описывается при помощи структур данных. Структуры данных конструируются из элементов данных и могут содержать альтернативы, условные вхождения и итерации. Условное вхождение означает, что данный компонент может отсутствовать в структуре. Альтернатива означает, что в структуру может входить один из перечисленных элементов. Итерация означает вхождение любого числа элементов в указанном диапазоне, Для каждого элемента данных может указываться его тип (непрерывные или дискретные данные). Для непрерывных данных может указываться единица измерения (кг, см и т.п.), диапазон значений, точность представления и форма физического кодирования. Для дискретных данных может указываться таблица допустимых значений.

Анализ и оценка производительности АСОИУ

Теоретические положения системного анализа определенное время рассматривались только как некая философия инженера и поэтому при решении задач создания искусственных систем иногда не учитывались. Однако развитие техники привело к тому, что без СА, одним из результатов которого являются концептуальные модели, исследование функционирования систем становится невозможным.

Первоначально компьютер отождествлялся с центральным процессором, основной и понятной характеристикой были быстродействие, измеряемое числом команд в единицу времени. Поэтому современные методики оценки отражают только возможности центрального процессора. В основе такой оценки лежит понятие производительности. При этом выделяют так называемое «чистое» процессорное время – период работы собственно процессора при выполнении внутренних операций и время ответа, включающее выполнение операций ввода-вывода, работу ОС и т.д.

Есть 2 показателя производительности процессов по «чистому» времени:

1. показатель производительности процессоров на операциях с данными целочисленного типа MIPS – отношение числа команд в программе к времени ее выполнения

2. показатель производительности процессоров на операциях с данными вещественного типа при все кажущейся простоте критерия оценки (чем > MIPS, тем быстрее выполняется программа) его использование затруднено вследствие нескольких причин:

1. процессоры разной архитектуры имеют различный набор команд

2. применение математических сопроцессоров и оптимизирующих компиляторов увеличивает производительность системы, но рейтинг MIPS может уменьшится, т.к. время выполнения команд для операций над данными с плавающей точкой значительно больше и за единицу времени м.б. выполнено меньшее число команд, нежели при выполнении соответствующих этим командам подпрограмм

3. научные приложения в основном связаны с интенсивными вычислениями над вещественными числами с плавающей точкой, коммерческие и офисные – с целочисленной арифметикой и обработкой транзакций БД. Графические приложения критичны и к вычислительным мощностям, и к параметрам графической подсистемы.

Еще более сложные проблемы появляются при необходимости оценок многопроцессорных систем. Такое положение привело к разработке и использованию ряда тестов, ориентированных на оценку вычислительных систем с учетом специфики их предполагаемого использования. Поэтому оценка процессоров с разной архитектурой основана на создании тестовой смеси из типовых операторов, влияющих на их производительность.

Управление проектом АСОИУ

Применение формализованных методов управления проектами позволяет более обоснованно определять цели инвестиций и оптимально планировать инвестиционную деятельность, более полно учитывать проектные риски, оптимизировать использование имеющихся ресурсов и избегать конфликтных ситуаций, контролировать исполнение составленного плана, анализировать фактические показатели и вносить своевременную коррекцию в ход работ, накапливать, анализировать и использовать в дальнейшем опыт реализованных проектов. Таким образом, система управления проектами является одним из важнейших компонентов всей системы управления организацией.

Система управления проектами представляет собой организационно-технологический комплекс методических, технических, программных и информационных средств, направленный на поддержку и повышение эффективности процессов планирования и управления проектом. В основе комплекса лежит *программное обеспечение календарного планирования.*

Основные преимущества использования системы управления проектами включают:

- централизованное хранение информации по графику работ, ресурсам и стоимости;
- возможности быстрого анализа влияния изменений в графике, ресурсном обеспечении и финансировании на план проекта;
- возможность распределенной поддержки и обновления данных в сетевом режиме;
- возможности автоматизированной генерации отчетов и графических диаграмм, разработки документации по проекту.

Как правило, универсальные системы календарного планирования обеспечивают следующий набор функциональных возможностей:

- средства визуального проектирования структуры работ проекта;
- средства планирования по методу критического пути;
- средства ресурсного планирования (описание, назначение и оптимизация загрузки ресурсов);
- возможности стоимостного анализа;
- средства контроля за ходом исполнения проекта;
- средства создания отчетов и графических диаграмм;

- средства организации групповой работы.

Программное обеспечение для управления проектами традиционно разделяется на профессиональные системы и системы для массового пользователя.

Профессиональные системы предоставляют гибкие средства реализации функций планирования и контроля, но требуют больших затрат времени на подготовку и анализ данных и соответственно высокой квалификации пользователей.

Системы для массового пользователя адресованы пользователям-непрофессионалам, для которых управление проектами не является основным видом деятельности. От пользователей, использующих пакеты планирования лишь время от времени при необходимости спланировать небольшой комплекс работ или ввести фактические данные по проекту, нельзя ожидать серьезных затрат времени и усилий на то, чтобы освоить и держать в памяти какие-либо специфические функции планирования или оптимизации расписаний. Для них более важным является простота использования и скорость получения результата.

В настоящее время даже относительно дешевые системы способны поддерживать планирование проектов, состоящих из десятков тысяч задач и использующих тысячи видов ресурсов. Основные различия между системами проявляются в реализации функций ресурсного планирования и многопроектного планирования и контроля. Возможности эффективного внедрения системы управления проектами во многом зависят от возможностей настройки системы на специфические показатели конкретных проектов, гибкости средств обмена данными, возможностей стандартизации управленческой среды и обеспечения групповой работы с данными проекта.

Microsoft Project и Time Line - недорогие системы управления проектами, просты в использовании, доступны для новичков и непрофессионалов. Они содержат базовые возможности, позволяющие осуществлять достаточно гибкое планирование и управление людскими ресурсами, поддерживают несложное многопроектное планирование и контроль.

MicrosoftProject - один из лидеров по возможностям объединения участников проекта средствами электронной почты или Интранет. При описании ресурса для каждого исполнителя может быть указан адрес его электронной почты. Информация о работах проекта может сохраняться в формате HTML и публиковаться на внутреннем Web-сервере. Кроме стандартных форматов файлов Microsoft Project (MPP и MPX) пользователь может сохранять информацию о проекте в форматах ODBC, Excel и Access. Формат MPD (Microsoft Project Database) позволяет хранить все данные о проекте в структуре, доступной как из Microsoft Project, так и из Access.

Что касается *TimeLine,* то система позволяет хранить все данные, касающиеся проектов организации, в единой базе данных. Отдельный модуль импорта/экспорта позволяет обмениваться данными с другими системами (Microsoft Project, Time Line for Windows), базами данных (dbf) и электронными таблицами. Система Time Line поддерживает стандарты ODBC, OLE, DDE и макроязык Symantec Basic.

Однако возможности недорогих систем, к которым относятся Microsoft Project и Time Line, не позволяют в полной мере реализовать режим многопользовательской работы с информацией проекта, поскольку не обеспечивают режим распределенного ввода данных и системы ограничения доступа к данным.

OpenPlanProfessional (Welcom Software) - представитель класса профессиональных систем. Одним из основных отличий системы являются мощные средства ресурсного и стоимостного планирования, которые позволяют значительно облегчить задачу нахождения наиболее эффективного распределения ресурсов и составления их рабочего расписания. Кроме того, пользователями интегрированной системы управления проектами организации являются как профессиональные менеджеры, осуществляющие согласование и оптимизацию планов проектов, анализ рисков, прогнозирование и т.д., так и участники проектов, выполняющие сбор, уточнение и актуализацию данных, готовящие отчеты. Ес-

ли для профессионалов важны мощность и гибкость предоставляемых системой функций планирования и анализа состояния проектов, то для остальных пользователей важнее простота и прозрачность системы. Open Plan обеспечивает как полную интеграцию между профессиональной и настольной версиями системы, так и открытость для обмена данными с внешними приложениями.

Open Plan поставляется в двух вариантах (Professional и Desktop), каждый из которых отвечает различным потребностям исполнителей, менеджеров и других участников проекта. Обе версии работают с одной базой данных. Совместное использование профессиональной и "облегченной" версий системы управления проектами дает возможность не только учесть потребности всех групп пользователей, но и значительно снизить стоимость решения.

Open Plan обладает прямым доступом к базам данных. Пользователь может выбрать, в каком формате хранить данные по проектам (в собственном формате Open Plan, в форматах Oracle, SQL Server, Sybase, xBase).

Open Plan обеспечивает ограничение доступа к данным проекта, предоставляя различные права на доступ к определенным данным, делая их доступными ограниченному кругу лиц и регулируя их совместное использование. Средство "Директор управления проектами", встроенное в Open Plan, позволяет упорядочить применение стандартных элементов проектов и процедур. В Open Plan предлагается 65 моделей, построенных на базе руководств PMI (Project Management Institute - Институт проектного менеджмента, США), которые можно настроить для создания документов, отвечающих требованиям стандартов ISO.

Проектная документация

Стадии создания

Стандарт распространяется на автоматизированные системы (АС), используемые в различных видах деятельности (исследование, проектирование, управление и т.п.), включая их сочетания, создаваемые в организациях, объединениях и на предприятиях (далее организациях). Стандарт устанавливает стадии и этапы создания АС, а также содержание работ на каждом этапе.

Процесс создания АС представляет собой совокупность упорядоченных во времени, взаимосвязанных, объединенных в стадии и этапы работ, выполнение которых необходимо и достаточно для создания АС, соответствующей заданным требованиям. Стадии и этапы создания АС в общем случае:

1. Формирование требований к АС;
2. Разработка концепции АС;
3. Техническое задание;
4. Эскизный проект;
5. Технический проект;
6. Рабочая документация;
7. Ввод в действие;
8. Сопровождение АС.

Допускается исключать стадию "Эскизный проект" и отдельные этапы работ на всех стадиях, объединять стадии "Технический проект" и "Рабочая документация" в одну стадию "Технорабочий проект". В зависимости от специфики создаваемых АС и условий их создания допускается выполнять отдельные этапы работ до завершения предшествующих стадий, параллельное во времени выполнение этапов работ, включение новых этапов работ.

Требования к содержанию документов

Настоящие методические указания распространяются на автоматизированные системы (АС), используемые в различных сферах деятельности (управление, исследование,

проектирование и т.п.), включая их сочетание, и устанавливают требования к содержанию документов, разрабатываемых при создании АС.

Содержание документов является общим для всех видов АС и, при необходимости, может дополняться разработчиком документов в зависимости от особенностей создаваемой АС. Допускается включать в документы дополнительные разделы и сведения, объединять и исключать разделы. Содержание каждого документа определяет разработчик в зависимости от объекта проектирования (системы, подсистема и т.д.).

1. Пояснительные записки к эскизному, техническому проектам содержат разделы: общие положения; описание процесса деятельности; основные технические решения; мероприятия по подготовке объекта автоматизации к вводу системы в действие.

2. Схема функциональной структуры содержит:

- элементы функциональной структуры АС (подсистемы АС); автоматизированные функции и/или задачи (комплексы задач); совокупности действий (операций), выполняемых при реализации автоматизированных функций только техническими средствами (автоматически) или только человеком;
- информационные связи между элементами и с внешней средой с кратким указанием содержания сообщений и/или сигналов, передаваемых по связям, и при необходимости, связи других типов (входимости, подчинения и т. д.);
- детализированные схемы частей функциональной структуры (при необходимости).

3. Описание автоматизируемых функций содержит разделы: исходные данные; цели АС и автоматизированные функции; характеристика функциональной структуры; типовые решения (при наличии).

4. Описание постановки задачи (комплекса задач) содержит разделы: характеристики комплекса задач; выходная информация; входная информация.

5. Общее описание системы содержит разделы: назначение системы; описание системы; описание взаимосвязей АС с другими системами; описание подсистем (при необходимости).

6. Программа и методика испытаний должна содержать перечни конкретных проверок (решаемых задач), которые следует осуществлять при испытаниях для подтверждения выполнения требований ТЗ, со ссылками на соответствующие методики (разделы методик) испытаний. Перечень проверок, подлежащих включению в программу испытаний, включает: соответствие системы ТЗ; комплектность системы; комплектность и качество документации; комплектность, достаточность состава и качество программных средств и программной документации; количество и квалификация обслуживающего персонала; степень выполнения требований функционального назначения системы; контролепригодность системы; выполнение требований техники безопасности, противопожарной безопасности, промышленной санитарии, эргономики; функционирование системы с применением программных средств.

Программа испытаний содержит разделы: объект испытаний; цель испытаний; общие положения; объем испытаний; условия и порядок проведения испытаний; материально-техническое обеспечение испытаний; метрологическое обеспечение испытаний; отчетность.

7. Схема организационной структуры содержит:

- состав подразделений (должностных лиц) организации, обеспечивающих функционирование АС либо использующих при принятии решения информацию, полученную от АС;
- основные функции и связи между подразделениями и отдельными должностными лицами, указанными на схеме, и их подчиненность.

8. Описание организационной структуры содержит разделы: изменения в организационной структуре управления объектом; организация подразделений; реорганизация существующих подразделений управления

9. Методика автоматизированного проектирования содержит разделы: общие положения; постановка задачи; методика проектирования; исходные данные; проектные процедуры; оценка результатов.

10. Перечень входных сигналов и данных содержит разделы: перечень входных сигналов; перечень входных данных.

11. Перечень выходных сигналов (документов) содержит разделы: перечень выходных сигналов; перечень выходных документов.

12. Описание информационного обеспечения системы содержит разделы: состав информационного обеспечения; организация информационного обеспечения; организация сбора и передачи информации; построение системы классификации и кодирования; организация внутримашинной информационной базы; организация внемашинной информационной базы.

13. Описание организации информационной базы содержит описание логической и физической структуры базы данных и состоит из двух частей: описание внутримашинной информационной базы; описание внемашинной информационной базы. Части документа содержат следующие разделы: логическая структура; физическая структура (для внутримашинной информационной базы); организация ведения информационной базы. В разделе "Логическая структура" приводят описание состава данных, их форматов и взаимосвязей между данными. В разделе "Физическая структура" приводят описание избранного варианта расположения данных на конкретных машинных носителях.

14. Описание систем классификации и кодирования содержит перечень применяемых в АС зарегистрированных классификаторов всех категорий по каждому классифицируемому объекту, описание метода кодирования, структуры и длины кода, указания о системе классификации и другие сведения по усмотрению разработчика.

15. Описание массива информации содержит: наименование массива; обозначение массива; наименование носителей информации; перечень реквизитов в порядке их следования в записях массива с указанием по каждому реквизиту: обозначения алфавита, длины в знаках и диапазона изменения (при необходимости), логических и семантических связей с другими реквизитами данной записи и другими записями массива; оценку объема массива; другие характеристики массива (при необходимости).

Документация по проекту:

Полный пакет включает следующие: пояснительная записка схема функциональной структуры; общее описание системы; описание автоматизируемых функций; описание постановки задачи; описание информационного обеспечения системы; описание организации инфо базы; перечень входных сигналов и данных; перечень выходных сигналов/документов; описание ПО; техническое задание; описание программы; пояснительная записка; программа и методика испытаний.

Инструментальные средства проектирования АСОИУ

Для реализации программных проектов используется широкий спектр современных инструментальных средств:

- CASE-средства - позволяющие значительно сократить сроки проектирования (за счет его организации в форме последовательной формализации и детализации знаний о проектируемой системе) и повысить качество проекта (за счет его представления в виде целостной модели, описывающей псе стороны функционирования автоматизируемого объекта); Westmount I-CASE фирмы CADRE, CASE/4/0 фирмы MICROTOOL, S-Dcsiirnor фирмы SYBASE (POWERSOFT). SilverRun фирмы CSA и др.
- 4С1-средства проектирования многоплатформенных приложений -позволяющие в минимальные сроки создавать прикладные программы, работающие практически на любых программно-аппаратных платформах: UNIFACE фирмы COMPUWARE, PowerBuilder фирмы SYBASE (POWERSOFT), Delphi фирмы BORLAND, JAM фирмы IYACC и др.,

- монитор транзакций TUXEDO фирмы BFA SYSTEMS для проектирования критически важных приложений реального времени (например, в банковской сфере),
- Internet / Intranet - средства проектирования: фирм ORACLE, NOVELL, MICROSOFT, BORLAND,
- системы управления базами данных: фирм ORACLE, INFORMIX, SYBASE, MICROSOFT, BTRIEVE,
- средства тестирования программного обеспечения, администрирования вычислительной среды и поддержки работы приложений (фирмы COMPUWARE),
- технологические решения третьих фирм, которые встраиваются в проект системы и значительно сокращают трудоемкость проектирования, повышают качество проекта: системы работы с изображениями, средства обеспечения коллективной работы, системы электронной почты и т. д.

Типизация проектных решений

Заказные системы.

При данном подходе вы заказываете разработку АСОИУ, как, например, заказывают нестандартную мебель. Это второй исторически сложившийся подход к построению АСОИУ. В «чистом виде» он предполагает разработку системы, полностью соответствующей особенностям конкретного предприятия, что и является его основным преимуществом. В потенциале этот подход характеризуется сравнительно меньшей стоимостью и меньшими сроками реализации, чем самостоятельная разработка.

В современных условиях при выборе этого подхода мы рекомендуем вам учитывать следующие его технологические и организационные особенности.

С технологической точки зрения наивно полагать, что разработчики будут создавать заказанную вами систему действительно «с нуля» (а если вдруг и будут, то это явный путь к провалу проекта). У них наверняка есть заранее наработанные решения, которые будут адаптироваться к вашим требованиям. Таким образом, во многих случаях сегодня «заказная» разработка фактически сводится к неявному использованию тиражируемых систем, которые имеются в распоряжении исполнителя. Результат разработки в этот случае во многом будет определяться качеством этих систем. Поэтому прежде чем остановиться на данном подходе имеет смысл внимательно познакомиться с возможностями построения АСОИУ с явным применением тиражируемых средств, поскольку эти варианты могут быть дешевле при той же функциональности и надежнее в связи с применением широко апробированных решений.

С организационной точки зрения этот подход может быть реализован двумя способами: создание временного коллектива разработчиков на вашем предприятии путем привлечения специалистов со стороны и заключение договора со специализированной фирмой. Несмотря на то что первый способ может оказаться существенно дешевле (программистов, ищущих дополнительные заработки, в России хватает), мы настоятельно не рекомендуем его использовать. После окончания разработки (даже при ее успешном завершении) вы скорее всего останетесь с системой «один на один», поскольку созданный временный коллектив может распасться или заняться другими (например, более выгодными для него материально) делами. Можно, конечно, иметь в виду создание этим коллективом на основе вашей системы тиражируемого продукта и последующее участие вашей фирмы в его продажах. Но это уже тема для совсем другого разговора.

В целом с учетом высказанных выше соображений использование подхода с разработкой заказных АСОИУ можно рекомендовать предприятиям с действительно уникальными особенностями бизнеса.

Тиражируемые (коробочные) продукты.

При использовании этого подхода вы приобретаете программы автоматизации различных видов хозяйственного учета так же, как например, офисные программы или

компьютерные игры. В ряде случаев программы поставляются в красочно оформленной упаковке (коробке), откуда, собственно, и пошло их название. В комплект поставки входит более или менее подробная инструкция по установке и эксплуатации программы, пользуясь которой в большинстве случаев можно достаточно быстро ввести эту программу в эксплуатацию.

Основными преимуществами подобного подхода являются сравнительно низкая стоимость программ, простота и, соответственно, небольшие сроки их освоения и, в ряде случаев, хороший сервис по сопровождению обновления версий программного обеспечения. Кроме того, продукты ведущих фирм тиражируются в больших количествах, следовательно можно ожидать, что они хорошо апробированы и не содержат часто проявляющихся ошибок.

Единственным недостатком такого подхода является то, что собственно АСОИУ с его помощью практически создать не удается. Это объясняется недостаточной функциональностью и масштабом «коробочных» продуктов, а также проблемами совместимости систем различных производителей.

Использование коробочных продуктов целесообразно для малых предприятий и на средних предприятиях на начальных стадиях автоматизации финансово-хозяйственной деятельности.

Адаптируемые интегрированные системы.

Подход к построению АСОИУ с применением адаптируемых интегрированных систем, которые, как мы уже говорили, появились на российском рынке во второй половине 90-х годов, удачно сочетает ряд преимуществ уже рассмотренных нами подходов и свободен от их основных недостатков. Это, на первый взгляд, не совсем очевидное обстоятельство объясняется особенностями построения адаптируемых интегрированных систем, которые, если не вдаваться в технические подробности, состоят в следующем.

Во-первых, как мы уже отмечали в обзоре развития систем автоматизации, основу адаптируемой интегрированной системы составляет тщательно проработанное и предназначенное для тиражирования программное ядро. Это ядро изначально функционально ориентировано на возможность обеспечения комплексной автоматизации управленческого и других видов учета, данные которых необходимы в АСОИУ. Таким образом наличие этого ядра с одной стороны в потенциале обеспечивает интегрированным системам такие преимущества тиражируемых систем, как использование апробированных решений и, с другой – устраняет недостаточный уровень функциональности и проблемы совместимости «коробочных» продуктов.

Во-вторых, адаптируемые интегрированные системы содержат гибкие средства настройки характеристик и возможностей создаваемой АСОИУ на особенности бизнеса конкретной организации. Поэтому при таком подходе к разработке АСОИУ появляется возможность удовлетворения требований заказчика, как это характерно для самостоятельно разрабатываемых или заказных систем, но сроки и риск неудачного выполнения работ здесь могут быть существенно сокращены за счет использования апробированного тиражируемого ядра.

В результате, АСОИУ, построенные с использованием этого подхода отличаются сравнительно небольшим временем разработки, эффективностью решения задач автоматизации управления и сравнительной простотой модификации при изменении организационной структуры предприятия или существующих бизнес-процессов.

Отмеченные преимущества подхода к построению АСОИУ с применением адаптируемых интегрированных систем позволяют рекомендовать его использование большинству из рассматриваемого в книге вида предприятий.

Как видно из приведенной выше схемы, адаптация интегрированной системы в принципе может производиться как силами заказчика, так и силами сторонней фирмы. Далеко не все из рассматриваемых в книге предприятий обладают специалистами, квалификация которых позволяет самостоятельно провести адаптацию, поскольку, как мы уви-

дим ниже, это достаточно емкий процесс, в котором необходимо учитывать целый ряд факторов. В рамках данного подхода для большинства из рассматриваемых в книге предприятий можно рекомендовать использование адаптируемых интегрированных систем с настройкой разработчиком или третьей (но обязательно специализированной) фирмой. Особенности построения АСОИУ на основе адаптируемых интегрированных систем мы рассмотрим в следующем разделе.

Адаптируемые интегрированные системы как платформа современных комплексных систем автоматизации.

Разработка АСОИУ на базе адаптируемых интегрированных систем ведется, как правило, на основе соответствующего договора между заказчиком и исполнителем. Содержание этих договоров при работе с различными поставщиками систем может, естественно, быть достаточно разнообразным. Однако практика показывает, что процесс создания АСОИУ на базе адаптируемых интегрированных систем в общем случае включает в себя несколько этапов.

Заключение договора. Это очень важный и ответственный этап, который в ряде случаев может определить успех или неудачу всей дальнейшей работы.

Обследование предприятия. Целью обследования является получение материалов для оценки эффективности управленческо-организационных механизмов компании, способов ведения работ, выполнения бизнес-процессов, принятых в конкретной компании, принятых схем учёта, используемой информации для принятия решений, т.е. общей технологии работы.

Проектирование модели бизнеса. Строится модель функционирования компании в том виде, в каком этот процесс должен быть после устранения проблем, обнаруженных на этапе обследования.

Консультации. Несмотря на то, что этот этап выделен у нас отдельным пунктом, консультации могут осуществляться на протяжении всего проекта, параллельно с другими работами, например, по вопросам постановки управленческого и других видов учета, минимизации налогообложения, юридическо-правовым нормам и т.п.

Настройка автоматизированной системы на модель бизнеса. На этом этапе происходит собственно адаптация интегрированной системы исходя из построенной модели бизнеса. Работа здесь выполняется в основном исполнителем. Заказчик может использовать это время для реорганизации управления в соответствии с построенной моделью бизнеса.

Технологическое внедрение. На этом этапе проводится установка у заказчика необходимого оборудования и соответствующих программ. Этап для крупных проектов может выполняться последовательно. Здесь же проводятся обучение пользователей работе с новой системой и ее опытная эксплуатация, по результатам которой могут вноситься определенные изменения в настройки системы. Этап завершается передачей системы в плановую эксплуатацию с оформлением соответствующих документов.

Сопровождение и развитие. В зависимости от условий договора на создание системы могут осуществляться необходимые консультации, поставка обновленных версий программных модулей, разработка при необходимости, новых функциональных модулей и т. д.

Отметим, что у различных поставщиков адаптируемых интегрированных систем наименования этапов и состав работ на этих этапах могут отличаться от приведенных выше. Однако суть дела при этом не меняется, хотя вопрос о полноте оказываемых исполнителем услуг при создании АСОИУ в практическом плане является весьма важным.

Графические средства представления проектных решений

Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

CASE-средства обладают следующими основными особенностями:

1. имеют мощные графические средства для описания и документирования ИС, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности;

2. осуществляют интеграцию отдельных компонент CASE-средств, обеспечивающую управляемость процессом разработки систем;

3. используют специальным образом организованное хранилище проектных метаданных (репозитория).

Интегрированное CASE-средство должно содержать следующие компоненты:

1. репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;

2. графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (DFD, ERD и др.), образующих модели ИС;

3. средства разработки приложений, включая языки 4GL и генераторы кодов;

4. средства конфигурационного управления;

5. средства документирования;

6. средства тестирования;

7. средства управления проектом;

8. средства реинжиниринга.

Современный рынок программных средств насчитывает около 300 различных CASE-средств, наиболее мощные из которых используются практически всеми ведущими западными фирмами.

Все современные CASE-средства могут быть классифицированы в основном по типам и категориям. Классификация по типам отражает функциональную их ориентацию на те или иные процессы ЖЦ.

Классификация по категориям определяет степень интегрированности по выполняемым функциям и включает следующее:

1. отдельные локальные средства, решающие небольшие автономные задачи (tools);

2. набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла систем (toolkit);

3. полностью интегрированные средства, поддерживающие весь ЖЦ систем и связанные общим репозиторием.

Помимо этого CASE-средства можно классифицировать по следующим признакам:

1. применяемым методологиям и моделям систем и БД;

2. степени интегрированности с СУБД;

3. доступным платформам.

Классификация по типам в основном совпадает с компонентным составом CASE-средств.

На сегодняшний день российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами: Vantage Team Builder (Westmount I-CASE), Designer/2000, Silverrun, ERwin+Bpwin, S-Designor, CASE-Аналитик, CASE /4/0, PRO-IV, System Architect, Visible Analyst Workbench, EasyCASE; VIS; RATIONAL ROSE.

Рассказать о UML, DFD, IDEF0

ERD – данная нотация используется в CASE средстве Oracle Designer.

Первый шаг моделирования – извлечение информации из интервью и выделение сущностей.

Второй шаг моделирования – идентификация связей. Связь это ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным количеством экземпляров, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности родителя. Имя связи между двумя сущностями должно быть уникальным. Имена связи модели недолжны, быть уникальны. Имя связи формируется с точки зрения родителя. Степень и обязательность связи можно показать графически.

Третий шаг – идентификация атрибута. Атрибут может быть либо обязательным, либо не обязательным. Каждый атрибут идентифицируется уникальным номером и изображается в виде списка имен внутри блока ассоциированной сущности, причем каждый атрибут занимает отдельную строчку. Каждая сущность обладает хотя бы одним возможным ключом.

Возможный ключ – один или несколько атрибутов, чьи значенья однозначно определяют каждый экземпляр сущности.

Супертипы и подтипы: одна сущность является обобщающим понятием для группы подобных сущностей.

Взаимно исключающие связи: каждый экземпляр сущности участвует только в одной связи из группы взаимно исключающих связей.

Рекурсивная связь – сущность может быть связана сама с собой.

Неперемещаемые связи – экземпляр сущность не может быть перенесен из одного экземпляра связи в другой.

2.Методические рекомендации (указания) к практическим занятиям

Состав рабочей документации.

Стадия рабочего проектирования заключается в составлении рабочей документации по комплексу программных и технических средств АСОИУ. Основанием для начала рабочего проектирования является утвержденный технический проект. При этом сам рабочий проект (как результат данной стадии) утверждению не подлежит. В процессе рабочего проектирования должен быть проведен комплекс мероприятий по подготовке объекта управления к внедрению АСОИУ: сформирован комплекс технических средств; завершено создание информационной базы; выполняется кодирование и отладка программ, настройка ППП; решены вопросы обучения пользователей и обслуживающего персонала; разработаны и утверждены технология обработки данных и должностные инструкции, определяющие порядок работы соответствующих специалистов в условиях применения АСОИУ. Стадии технического и рабочего проектирования могут быть совмещены в том случае, когда имеются проверенные эффективные проектные решения по основным видам обеспечения АСОИУ, пригодные для применения в конкретном проекте.

Опишите стадию рабочего проектирования АСОИУ для ремонтного предприятия, ее назначение и перечень работ.

Принципы построения функциональных подсистем.

Функциональные подсистемы КИС могут строиться по различным принципам:
• проблемному;
• предметному;
• функциональному;
• смешанному (предметно-функциональному).

Проблемный принцип формирования подсистем отражает необходимость гибкого и оперативного принятия управленческих решений по отдельным проблемам в рамках си-

стем поддержки принятия решений, например, решение задач бизнес-планирования, управления проектами. Такие подсистемы можно реализовывать в виде локальных ИС, импортирующих данные из КИС (например, система бизнес-планирования на основе ППП *Project-Expert*), или в виде специальных подсистем в рамках КИС (например, ИС руководителя).

На практике чаще всего применяют **смешанный предметно-функциональный** подход, согласно которому построение функциональной структуры КИС – это разделение ее на подсистемы по характеру хозяйственной деятельности, которое должно соответствовать структуре объема и системе управления, а также характеру выполняемых функций управления. При этом подходе можно выделить следующий типовой набор функциональных подсистем в общей структуре КИС предприятия:

Функциональный принцип:

• перспективное развитие;

• технико-экономическое планирование;

• бухгалтерский учет и анализ хозяйственной деятельности.

Предметный принцип (подсистемы управления ресурсами):

• техническая подготовка производства;

• управление основным производством;

• управление вспомогательным производством;

• управление качеством продукции;

• управление материально-техническим снабжением;

• управление реализацией и сбытом готовой продукции;

• управление кадрами.

Подсистемы, построенные по функциональному принципу, охватывают все виды хозяйственной деятельности предприятия (производство, снабжение, сбыт, персонал, финансы). Подсистемы, построенные по предметному принципу, относятся в основном к оперативному уровню управления ресурсами.

С учетом предметной направленности использования АСОИУ в хозяйственных процессах промышленного предприятия можно выделить подсистемы, соответствующие управлению отдельными ресурсами (управление есть процесс целенаправленной переработки информации):

• управление сбытом готовой продукции;

• управление производством;

• управление материально-техническим снабжением;

• управление финансами;

• управление персоналом.

При этом в подсистемах рассматривается решение задач на всех уровнях управления, обеспечивая интеграцию информационных потоков по вертикали

Опишите принципы построения функциональных подсистем для ремонтного предприятия.

Критерии выбора обеспечивающих подсистем информационных систем.

Предлагаемые на рынке информационные системы обладают достаточной гибкостью благодаря модульной архитектуре. В результате этого ИТ-директор сталкивается с необходимостью решить задачу, какие бизнес-функции ему необходимо реализовать сейчас, какие заложить на реализацию в будущем, а от каких необходимо и вовсе отказаться. То есть важно расставить приоритеты в функциональности ИС и запланировать ее развитие в будущем. В итоге получается целый набор вариантов ИТ-решений, из которого необходимо выбрать наиболее оптимальный как с точки зрения текущей функциональности, так и с точки зрения стратегического развития ИТ-комплекса предприятия в целом и оптимальности финансовых затрат.

Также стоит обратить внимание на подготовку качественной тендерной документацию, которая позволит ИТ-компаниям сделать предложения по срокам и стоимости проекта без проведения предварительного обследования. Компания может подготовить тендерную документацию самостоятельно или привлечь консалтинговую компанию. Для подготовки тендерной документации необходимо описать бизнес-процессы компании «как должно быть» («to be»), то есть спроектировать новые процессы и улучшить существующие. Эта работа может быть эффективно выполнена с привлечением консультантов, особенно в тех организациях, где отсутствует собственная экспертиза в области моделирования бизнес-процессов и формализации требований к информационной системе.

Основные критерии выбора информационных систем:

1. Масштабируемость – подразумевает возможность увеличить необходимую производительность системы как по количеству операций, так и по числу пользователей.
2. Надежность и безопасность – устойчивость системы к сбоям.
3. Сопровождаемость – информационная система не должна отнимать слишком много ресурсов на свое обслуживание.
4. Стандарты – система должна быть построена по современным стандартам открытых систем (например, стандарты APICS).
5. Производительность.
6. Дружественный интерфейс пользователя.

Критерии по выбору платформы для хранилища данных более детальны. Существует несколько ключевых факторов, осложняющих обеспечение интегрированной информацией в крупных организациях. Для того чтобы преодолеть эти осложняющие факторы, любое решение для корпоративного хранилища данных должно удовлетворять следующим требованиям:

1. Обеспечивать определенную скорость доступа и гибкость, что включает:

· Итеративный подход к внедрению, позволяющий учитывать меняющиеся бизнес-требования. Возможность быстрого обеспечения решения для конкретной области бизнеса, а также дальнейших расширений.

· Возможность быстрого создания прототипов, чтобы новые бизнес-требования могли быть оперативно изучены вместе с пользователями.

· Отсутствие жестких схем, т.к. пользователи часто обнаруживают, что им нужна совсем другая информация из хранилища, чем они первоначально предполагали.

2. Обеспечивать масштабируемость:

· Подход должен быть масштабируемым, для того чтобы общий проект мог быть разделен на несколько отдельных (более мелких) проектов, направленных на удовлетворение различных приоритетов, имеющихся в тех или иных подразделениях организации. Это также упростит общее внедрение, т.к. позволит избежать осуществления очень крупных проектов.

· Хранилище должно легко расширяться для включения новых сфер бизнеса или наборов данных транзакций. Например, должна существовать возможность сначала создать хранилище для областей продаж и маркетинга, а затем включить в него сферу финансов.

3. Обеспечивать расширяемость:

· По мере развития, роста и изменения бизнеса неизбежно появляются потенциальные новые источники данных (например, при приобретении другой компании). Очень важно иметь возможность быстро включить эти новые источники данных в хранилище без сложной и длительной реконструкции последнего.

4. Обеспечивать ориентацию на пользователей:

· Часто хранилище может использоваться как цельный источник данных для создания различных подмножеств данных, таких как витрины или кубы, что облегчает разработку отчетности. Хранилище должно поддерживать процесс создания таких витрин и помогать пользователю в извлечении наборов данных.

· Конструктивное решение хранилища данных должно позволять бизнес-пользователям принимать участие как в первоначальной разработке хранилища, так и в последующих изменениях его дизайна, связанных с изменениями требований бизнеса. Как показывает опыт, чем больше вовлечены пользователи в общее проектирование хранилища, тем более вероятно, что оно будет удовлетворять требованиям бизнеса. Например, в хранилище должен учитываться тот факт, что один и тот же субъект бизнеса в одних случаях может рассматриваться как клиент, а в других – как поставщик.

· Хранилище данных должно быть легко адаптируемым, чтобы пользователи могли работать только на уровне бизнеса, не осложняя себе жизнь, например, названиями таблиц. Также изменения в хранилище, связанные с изменениями в бизнесе, не должны требовать выгрузки всех данных и их повторной последующей загрузки. Это замедляет развитие бизнеса и затрудняет способность быстро реагировать на изменения.

5. Обеспечивать способность реагировать на изменения и осуществлять бизнес-моделирование:

· Хранилище данных должно обладать способностью учитывать временную зависимость и вариабельность, т.е. поддерживать так называемую "корпоративную память": информацию о прошлом, настоящем и будущем бизнеса. Организации хотят иметь такие хранилища, которые могут функционировать в течение длительного периода времени и хранить историческую информацию. В течение этого периода некоторые объекты могут измениться. Если эти исторические данные не сохраняются, то сравнение состояний одного и того же объекта во времени окажется невозможным.

· Необходима функция поддержки планирования бизнес-сценариев, т.е. способность рассматривать и анализировать данные в различных аспектах, включая исторические тенденции. Это необходимо для изучения будущих возможностей бизнеса. Помимо этого, часто возникают новые сценарии и структуры отчетности, которые должны быть включены в хранилище до того, как они вступят в силу. Это также важно для планирования различных сценариев.

Опишите выбор обеспечивающих подсистем информационных систем для ремонтного предприятия.

Принципы проектирования документооборота.

Любой компании, занимающейся разработкой и верификацией программного обеспечения (ПО), необходимо хранить и обрабатывать большое количество информации в электронном виде. Чем сложнее проекты и чем выше их критичность, тем выше требования к процессу управления данными. Более того, для обеспечения гарантии качества выходного продукта необходимо установить, а затем поддерживать определенные процессы, регламентируемые как внутренними, так и отраслевыми стандартами. Это особенно важно для таких областей, как встроенное ПО, где цена ошибки может быть очень велика или даже сопряжена с угрозой жизни людей.

Для структурирования, управления и анализа электронной информации используют различные виды систем документооборота, которые оперируют термином «документ». (Под документом следует понимать некий набор структурированной информации, в общем случае – произвольный объект. Такой объект обладает набором свойств, набором ролей, которые могут работать с документом, и жизненным циклом, описывающим все стадии и процессы работы с этим объектом.) В компании, связанной с разработкой и тестированием ПО, рано или поздно возникает необходимость внедрения системы документооборота. Такую систему можно специально разработать для конкретной компании или использовать уже готовую, предварительно внеся в нее все необходимые настройки. В любом случае система поддержи документооборота – это система для предприятия, поэтому все требования к ее функционированию заложены в его бизнес-процессах. А значит, пер-

вичная задача – выявление и описание всех документов будущей системы в соответствии с основными процессами компании.

Наиболее частая проблема автоматизации – неверное понимание потребностей предприятия разработчиками и, следовательно, предоставление неадекватной системы. Поэтому очень важно точно определить первичные требования и поддерживать тесное взаимодействие разработчиков с заказчиками и специалистами в области бизнес-процессов компании на протяжении всего жизненного цикла (ЖЦ) системы.

Проектируемая система должна работать с документами, которые проще представить в виде объектов. Поэтому наиболее подходящее и удобное средство построения модели документов предприятия и схем их функционирования – это широко распространенный язык объектного проектирования UML. Язык UML создавался с целью построения модели архитектуры именно программной системы, поэтому по таким моделям достаточно просто проводить реализацию.

Но начинать автоматизацию с документов и построения программной модели в качестве первичной непросто. Крупное предприятие с большим количеством разных бизнес-процессов и, возможно, с множеством филиалов, субподрядчиков и поставщиков удобно описывать, отталкиваясь от его бизнес-процессов, путем постепенной декомпозиции. Такой подход будет удобен и понятен для специалистов в области постановки и оптимизации бизнес-процессов, а именно, они должны определить первичные требования к системе документооборота. Наиболее простой и удобной методологией описания процессов являются SADT и языки IDEF0 и IDEF3. При помощи этой методологии можно описать функционирование больших организаций и декомпозировать все процессы до понятных и простых функций.

Предлагаемый авторами подход заключается в объединении двух методов проектирования: объектного UML и процессного IDEF. Описание структур документов и их динамического поведения строится на основе моделей бизнес-процессов предприятия. Каждый документ в данном случае представляет собой набор данных, задающих его структуру (атрибуты, заполняемые информацией) и модель ЖЦ, определяющего правила работы с документом, изменения его состояния и правила разграничения доступа. Каждый документ в системе представляет собой активный объект, отражающий бизнес-процессы предприятия в собственном ЖЦ. Система поддержки документооборота не содержит никаких схем процессов, она лишь интерпретирует описания документов, которые непосредственно функционируют в ее среде.

Сначала с использованием структурного подхода и нотации IDFE0 строится статическое функциональное описание процессов предприятия. Такое описание может уже существовать на момент проектирования системы документооборота, так как некоторые стандарты, например ISO 9001:2000, прямо ориентированы на его наличие в том или ином виде. При его отсутствии можно базироваться на типовых схемах бизнес-процессов компаний-разработчиков ПО. Далее эта модель дополняется динамической частью с помощью языка IDEF3. Затем по этим описаниям строится объектная модель функционирования документов, базирующаяся на изменении их состояний. Рассматриваемый метод построения системы включает в себя следующие аспекты:

- выделение основных типов документов (объектов);
- определение ЖЦ для выделенных типов документов и упрощение этих ЖЦ;
- задание ролей и прав для всех типов документов.

Выделение основных объектов производится путем объединения дуг (предметов) в IDEF0 модели с учетом терминологического словаря, семантики диаграмм и операций, в которых участвуют те или иные дуги.

В процессе моделирования обычно участвуют несколько разработчиков, поэтому неизбежно возникают расхождения в употреблении одних и тех же терминов, что может искажать смысл диаграмм. Для решения этой проблемы предлагается использовать тер-

минологический словарь, который должен быть единым в течение всего процесса функционального и объектного моделирования.

Опишите принципы проектирования документооборота для ремонтного предприятия.

Состав и содержание операций проектирования классификаторов.

Все классификаторы, разрабатываемые и используемые в АСОИУ, имеют эталонную и рабочую *формы. Эталонная форма классификатора* - это официальное издание классификатора на бумажном носителе, удобное для осуществления его ведения. *Рабо*чая *форма классификатора* - это весь классификатор или его раздел, занесенный на машинный носитель и удобный для обработки информации.

На первом этапе *«Разработка ТЗ на проектирование»* выполняются две работы. Первая из них связана с определением состава, назначения и сферы действия классификаторов, используемых в системе. *Перечень классификаторов* определяется на основе анализа реквизитного состава первичных и результатных документов и выделения всей совокупности реквизитов-признаков.

Далее определяют *назначение классификаторов.* Каждый классификатор может быть предназначен для однозначной идентификации объекта, передачи информации на расстояние по каналам связи или для поиска и логической обработки первичной информации с целью получения и выдачи результатной информации.

По *сфере действия* выделяют следующие виды классификаторов: *международные*, *общегосударственные* (общесистемные), *отраслевые* и *локальные* классификаторы.

Международные классификаторы входят в состав Системы международных экономических стандартов (*СМЭС*) и обязательны для передачи информации между организациями разных стран мирового сообщества. *СМЭС* представляет собой множество стандартных решений по классификационным группировкам и кодированию специальной и экономической информации и формированию источников этой информации. В состав *СМЭС* входят классификации Организации Объединенных наций (*ООН*) и ее специализированных образований, в том числе:

· Международная стандартная отраслевая классификация всех видов экономической деятельности (*МСОК*);

· Классификация основных продуктов (*КОП*);

· Международная стандартная торговая классификация (*МСТК*);

· Классификация по широким экономическим категориям (*КШЭК*);

· Классификация функций органов управления (*КФОУ*);

· Классификация функций правительства;

· Классификации продовольственных и сельскохозяйственных организаций (*РАО*);

· Классификации международной организации труда (МОТ);

· Классификации *ООН* по вопросам образования, науки и культуры (*ЮНЕСКО*);

· Международная стандартная классификация образования (*МСКО*).

К числу классификаций, разработанных другими международными организациями, относятся:

· Классификация международного энергетического агентства (*МЭА*);

· Классификация Совета таможенного сотрудничества;

· Система описания и кодирования товаров;

· Классификация Всемирной туристической организации (*ВТО*);

· Классификация деятельности, связанной с туризмом.

К классификациям Европейского сообщества и других международных региональных организаций относят:

· Классификацию Европейского сообщества (*ЕС*);

· Общую отраслевую классификацию экономической деятельности в рамках *ЕС* (*КДЕС*) и другие.

Второй класс классификаторов образуют *общегосударственные (общесистемные)* классификаторы, обязательные для организации процессов передачи и обработки информации между экономическими системами государственного уровня внутри страны, состав которых будет рассмотрен в следующем разделе.

Для выполнения процедур обработки информации и передачи ее между организациями внутри отрасли используют *отраслевые классификаторы.* В пределах отдельных предприятий используют, как правило, *локальные классификаторы.*

Выполнение работ второго этапа связано с *определением состава исходных данных и требований к разрабатываемым классификаторам.*

К *исходным данным,* используемым в процессе проектирования классификаторов, относятся:

· состав задач, для которых разрабатывается классификатор;

· состав объектов классификации и мощность исходного множества;

· состав признаков классификации и число значений каждого признака;

· наименования отдельных группировок и объектов;

· динамика процесса изменяемости состава задач, объектов и признаков.

К требованиям, которым должны удовлетворять разрабатываемые классификаторы, можно отнести:

· полноту охвата объектов и признаков классификации каждым классификатором;

· согласованность признаков деления множеств объектов с алгоритмами обработки экономической информации;

· взаимную однозначность наименований объектов и их кодовых обозначений;

· простоту кодирования и возможность автоматизации классификации и кодирования;

· возможность увязки с другими классификаторами и системами обозначений;

· эффективность использования классификатора при обработке информации.

Содержанием второго этапа является *«Разработка методических материалов проектирования»,* которая включает прежде всего *разработку основных критериев и принципов построения каждого классификатора.* К *критериям* построения классификатора относятся такие, как:

· критерий отнесения того или иного объекта к конкретному классифицируемому множеству;

· степень охвата кодируемого множества объектов.

Принципы построения классификатора определяются структурой классификатора, т.е. количеством ветвей, выходящих из каждой классификационной группировки, количеством ступеней и числом уровней классификации. Классификатор считается *однородным,* если на каждой ступени из каждой классификационной группировки выходит одинаковое количество ветвей.

Кроме того, на этом этапе разрабатывается *система взаимодействия* классификаторов разных уровней, предназначенных обеспечивать взаимодействие ЭИС с внешней средой. Эта работа представляет собой разработку некоторого транслятора перехода от одного классификатора к другому. Но чтобы его создать, необходимо провести выбор некоторой системы взаимодействия различных классификаторов, ориентированных на некоторую номенклатуру объектов. Существуют следующие системы взаимодействия:

а) *система равноправных классификаторов,* характеризуется тем, что на каждом уровне управления для целей обработки информации используется свой локальный классификатор, а для получения или передачи информации из внешней среды используется соответствующий транслятор. Недостаток данной системы заключается в том, что та система, которая имеет на входе наибольшее количество потоков информации от различных организаций, должна иметь наибольшее количество трансляторов;

б) *система приоритетных классификаторов,* применяется для предприятий одной отрасли. При этой системе на каждом предприятии этой отрасли и на каждом уровне управления имеются локальные классификаторы. Обмен информацией осуществляется в терминах классификатора вышестоящего уровня. Эта система дает уменьшение количества трансляторов независимо от числа входных и выходных потоков. Однако трудности возникают при передаче потоков информации между предприятиями, относящимися к разным отраслям;

в) *система классификаторов-посредников,* применяется при межотраслевом управлении. На каждом объекте любого уровня управления обработка ведется в терминах своего локального классификатора, а обмен - в терминах одного классификатора-посредника. Преимущества такой системы заключаются в необходимости создания только одного транслятора для каждого предприятия и в обеспечении возможности централизованного ведения классификатора-посредника, что дает минимальное количество ошибок при кодировании информации и обеспечивает информационную совместимость ЭИС разных уровней;

г) *система единого классификатора,* для обработки информации на всех предприятиях, входящих в состав экономической макросистемы, и для передачи этой информации между ними возможна только гипотетически, но реально ее нельзя осуществить из-за необходимости кодирования всей информации, существующей в стране, используя очень громоздкие классификаторы.

К *факторам,* влияющим на выбор способа увязки классификатора, относятся следующие:

· объем и характер обрабатываемой информации;

· объем и характер потоков получаемой и передаваемой информации;

· минимум трудовых и стоимостных затрат на разработку и эксплуатацию системы ведения классификаторов.

На этом же этапе осуществляется *разработка методик построения классификаторов,* отражающих методы и последовательность выполнения отдельных операций по созданию классификаторов, содержание которых зависит от выбранных критериев и принципов их построения.

Третий этап *связан с работами по организации сбора и обработки исходных данных,* необходимых для составления классификаторов. К их числу относится *разработка инструктивных материалов* по сбору и обработке исходных данных:

· определение перечня решаемых задач, использующих классификаторы;

· выделение классифицируемых объектов;

· определение состава признаков классификации и значений признаков;

· осуществление лингвистической обработки этих данных (удаление синонимов, омонимов, полисемии, антонимов и др.);

· согласование используемой терминологии в исходных данных с ГОСТами.

Другой работой, которую выполняют на этом этапе, являются *сбор и обработка данных* согласно разработанным инструкциям.

На четвертом этапе *«Составление классификаторов и системы их ведения»* осуществляется построение эталонной и рабочей формы классификатора и системы ведения классификатора.

Эталонный классификатор должен быть согласован, отпечатан типографским способом и распространен всем пользователям для кодирования информации первичных документов.

Рабочие классификаторы наносятся на машинные носители в необходимых разрезах, передаются пользователям и заносятся в файлы справочников баз данных для выполнения процедуры автоматического заполнения машинных форм первичных документов и для декодирования результатной информации, получаемой после ее обработки.

К задачам, решаемым *системой ведения классификатора,* относятся:

· актуализация классификатора, т.е. постоянное пополнение и изменение объектов классификации и кодирования;

· своевременное оповещение всех пользователей обо всех происходящих изменениях;

· реструктуризация, или пересмотр структуры классификатора, при котором осуществляются контроль за дублированием объектов классификации, контроль и выявление тупиковых ветвей, не ведущих к объекту, оптимизация резервных ветвей по всем уровням иерархии или по всем аспектам классификации;

К проблемам, связанным с проектированием системы ведения классификаторов, относятся:

· разработка организационной структуры системы ведения, т.е. службы, которая отвечает за пополнение классификаторов;

· разработка юридических основ внесения изменений в классификатор;

· разработка информационного и программного обеспечения системы.

Все работы по проектированию классификаторов заканчиваются *экспериментальной проверкой* и внесением коррективов, *утверждением, изданием* и *рассылкой* классификаторов всем пользователям в функциональные подсистемы.

Спроектируйте классификатор для ремонтного предприятия.

Требования к организации хранения файлов в информационной базе.

Хранение и накопление являются одними из основных действий, осуществляемых над информацией и главным средством обеспечения ее доступности в течение некоторого промежутка времени.

База данных может быть определена как совокупность взаимосвязанных данных, используемых несколькими пользователями и хранящихся с регулируемой избыточностью.

Банк данных – система, представляющая определенные услуги по хранению и поиску данных определенной группе пользователей по определенной тематике.

Система баз данных – совокупность управляющей системы, прикладного программного обеспечения, базы данных, операционной системы и технических средств, обеспечивающих информационное обслуживание пользователей.

Хранилище данных (ХД – используют также термины Data Warehouse, «склад данных», «информационное хранилище») – это база, хранящая данные, агрегированные по многим измерениям.

Альтернативой хранилищу данных является концепция витрин данных (Data Mart).

Витрины данных – множество тематических БД, содержащих информацию, относящуюся к отдельным информационным аспектам предметной области.

Еще одним важным направлением развития баз данных являются репозиторий. Репозиторий, в упрощенном виде, можно рассматривать просто как базу данных, предназначенную для хранения не пользовательских, а системных данных. Каждый из участников действия (пользователь, группа пользователей, «физическая память») имеет свое представление об информации.

По отношению к пользователям применяют трехуровневое представление для описания предметной области: концептуальное, логическое и внутреннее (физическое)

Концептуальный уровень связан с частным представлением данных группы пользователей в виде внешней схемы, объединяемых общностью используемой информации. Каждый конкретный пользователь работает с частью БД и представляет ее в виде внешней модели. Этот уровень характеризуется разнообразием используемых моделей (модель «сущность –связь», ER-модель, модель Чена), бинарные и инфологические модели, семантические сети.

Логический уровень является обобщенным представлением данных всех пользователей в абстрактной форме. Используются три вида моделей: иерархические, сетевые и реляционные.

Сетевая модель является моделью объектов-связей, допускающей только бинарные связи «многие к одному» и использует для описания модель ориентированных графов.

Иерархическая модель является разновидностью сетевой, являющейся совокупностью деревьев (лесом).

Реляционная модель использует представление данных в виде таблиц (реляций), в ее основе лежит математическое понятие теоретико-множественного отношения, она базируется реляционной алгебре и теории отношений.

Контрольные вопросы

1. Укажите отличия базы данных, хранилища данных, витрины данных, репозитария.
2. Какие модели используются для описания предметной области?
3. Какие модели используются на концептуальном уровне?
4. Какие модели используются на физическом уровне?
5. Дайте краткую характеристику основных типов баз данных

Задачи, определяемые опытной эксплуатацией.

Для АСОИУ устанавливают следующие основные виды испытаний:

1) предварительные (автономные или комплексные) проводят для определения ее работоспособности и решения вопроса о возможности приемки АСОИУ в опытную эксплуатацию;

2) опытная эксплуатация проводят с целью определения фактических значений количественных и качественных характеристик АСОИУ и готовности персонала к работе в условиях функционирования АСОИУ, определения фактической эффективности АСОИУ, корректировке (при необходимости) документации;

3) приемочные проводят для определения соответствия АСОИУ техническому заданию, оценки качества опытной эксплуатации и решения вопроса о возможности приемки АСОИУ в постоянную эксплуатацию.

4) Ниже перечислены организационно – распорядительные документы, оформляемых на этих стадиях:

5) акт завершения работ;

6) акт приемки в опытную эксплуатацию;

7) акт приемки в промышленную эксплуатацию;

8) Документ "Приказ о начале опытной эксплуатации АСОИУ (ее частей)"

9) Документ "Приказ о вводе в промышленную эксплуатацию АСОИУ (ее частей)"

10) Приказ о составе приемочной комиссии

11) Протокол испытаний

12) Протокол согласования.

Опишите задачи, определяемые опытной эксплуатацией для ремонтного предприятия.

Методологии проектирования АСОИУ.

Методы проектирования можно классифицировать следующим образом:

· по степени автоматизации

· ручное проектирование (без использования инструментальных средств)

· компьютерное (используются специальные среды, которые позволяют нам реализовывать проектные решения)

· по степени использования типовых проектных решений

· оригинальное проектирование (когда проектные решения разрабатываются с 0, в соответствии с требованиями АСОИУ)

· типовое проектирование (предполагающее конфигурацию АСОИУ из готовых типовых решений)

Сочетание различных методов проектирования обуславливает характер используемой технологии, в связи с чем выделают

· индустриальное проектирование (подразумевает использование различных case-технологий)

· каноническое проектирование (ручное проектирование)

Для конкретных видов технологии проектирования характерно применение определенных средств разработки АСОИУ, которые поддерживают выполнение как отдельных проектных работ, так и этапов проектирования в целом. Поэтому перед разработчиками АСОИУ стоит задача выбора таких средств, которые по своим характеристикам, отвечают требованиям конкретного предприятия.

Существуют две методологии проектирования:

1. Методология нисходящего проектирования (от целей проектирования к исходным данным) является основным при разработке сложных систем;

2. Методология восходящего проектирования (разворачивание проектирования от исходных данных к цели).

Сделайте выбор в пользу той или иной методологии проектирования АСОИУ для ремонтного предприятия.

3.Методические рекомендации (указания) к лабораторным работам

1. Программное средство структурного моделирования процессов RAMUS Educational.

Аннотация: Цель занятия: освоить интерфейс ИС РАМУС для моделирования БП в нотации IDEF0.

Ключевые слова: actor, communications, DLT, education, IDEF, KVM, UML, администрирование сетей, банковская выписка, валидация, детализация, единицы, жизненный цикл продукта, загрязнение окружающей среды, имя связи, кадровая политика, лингвистическое обеспечение, менеджерам, нумерация, операции, перемещение товаров, работ, связь, таблица, фактическая себестоимость

В результате выполнения заданий студент получит навык создания и редактирования функциональных моделей в программной среде Ramus. Работа предполагает последовательное выполнение заданий, поэтому необходимо сохранять модели, полученные *по* результатам каждого упражнения.

В качестве примера рассматривается *деятельность* промышленной компании. Компания занимается сборкой и продажей настольных компьютеров и ноутбуков. Компания не производит компоненты самостоятельно, а только собирает и тестирует компьютеры.

Деятельность компании состоит из следующих элементов:

- продавцы принимают заказы клиентов;
- операторы группируют заказы по типам клиентов;
- операторы собирают и тестируют компьютеры;
- операторы упаковывают компьютеры согласно заказам;
- кладовщик отгружает клиентам заказ.

Компания использует приобретенную бухгалтерскую ИС, которая позволяет оформить заказ, счет и отследить платежи *по* счетам.

Перед выполнением упражнения 1. Запустите программу Ramus (**Пуск -> Программы -> Ramus -> Ramus**). Если *программа* не установлена на ПК, то при наличии до-

ступа в *Интернет* самостоятельно произведите инсталляцию данного *ПО* с сайта разработчика: http://ramussoftware.com/.

Упражнение 1. Создание контекстной диаграммы

1. После запуска программы на экране появится окно начала работ (рис. 1). Выберите опцию "**Создать**" и нажмите "**ОК**".

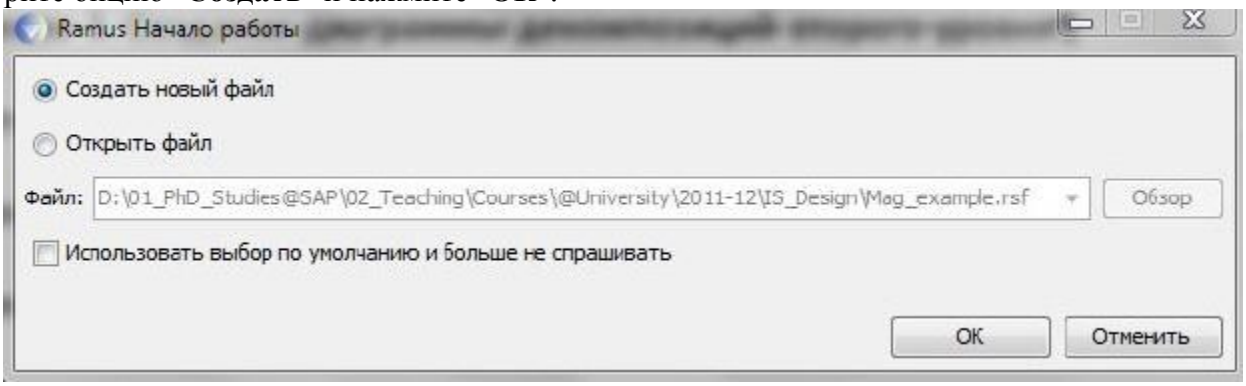


Рис. 1. Диалоговое окно начала работы в Ramus.

2. Внесите имя автора, название проекта, название модели и выберите опцию "**IDEF0**". На следующем шаге укажите, что модель используется "**отделом стратегического планирования и развития**".

В описании проекта укажите "**Это учебная модель, описывающая деятельность компании**", перейдите к следующему шагу.

3. Раздел "**классификаторы**" оставьте незаполненным и нажмите "**Дальше**".

4. В следующем диалоговом окне нажмите "**Окончить**" и перейдите к *рабочему интерфейсу программы*.

5. Через меню **Диаграмма -> Свойства модели** можно отредактировать метаданные модели, а именно: название модели, описание, место ее использования.

6. Активируйте окно модели, кликнув на область моделирования. Создайте *контекстную диаграмму*.

7. Перейдите в режим редактирования *контекстной диаграммы*, нажав правой кнопкой мыши на объекте и выбрав опцию "**Редактировать активный элемент**". В закладке "**Название**" введите "**Деятельность компании**". Во вкладке "**Описание**" введите "**Текущие бизнес-процессы компании**". Обратите внимание, что вкладка "**Описание**" может быть недоступна в версии RAMUS *Educational*

8. Создайте стрелки на контекстной диаграмме в соответствие с информацией, приведенной в таблице 1. Для создания стрелок необходимо перейти в режим построения стрелок, навести курсор на исходную точку стрелки (левая, верхняя и нижняя граница области построения модели или правая граница *контекстной диаграммы*), после того, как область будет подсвечена черным цветом, кликнуть один раз и аналогичным образом обозначить конец стрелки (правая, верхняя и нижняя граница *контекстной диаграммы* или правая граница области построения модели). Перемещать стрелки и их названия можно по принципам стандартного механизма drag&drop.

Таблица 1. Описание стрелок *контекстной диаграммы*

Название	**«Смысловая нагрузка»**	**Тип**
Бухгалтерская система	Оформление счетов, оплата, работа с заказами	Механизм
Звонки клиентов	Запросы информации, заказы, техническая поддержка и т.д.	Вход
Правила и процедуры	Правила продаж, инструкции по	Управляющие воз-

Название	«Смысловая нагрузка»	Тип
	сборке, процедуры тестирования, критерии производительности и т.д.	действия
Проданные продукты	Настольные и портативные компьютеры	Выход

На рис. 2 представлен результат построения *контекстной диаграммы по* результатам Упражнения 1.

На рис. 2 представлен результат построения *контекстной диаграммы по* результатам Упражнения 1.

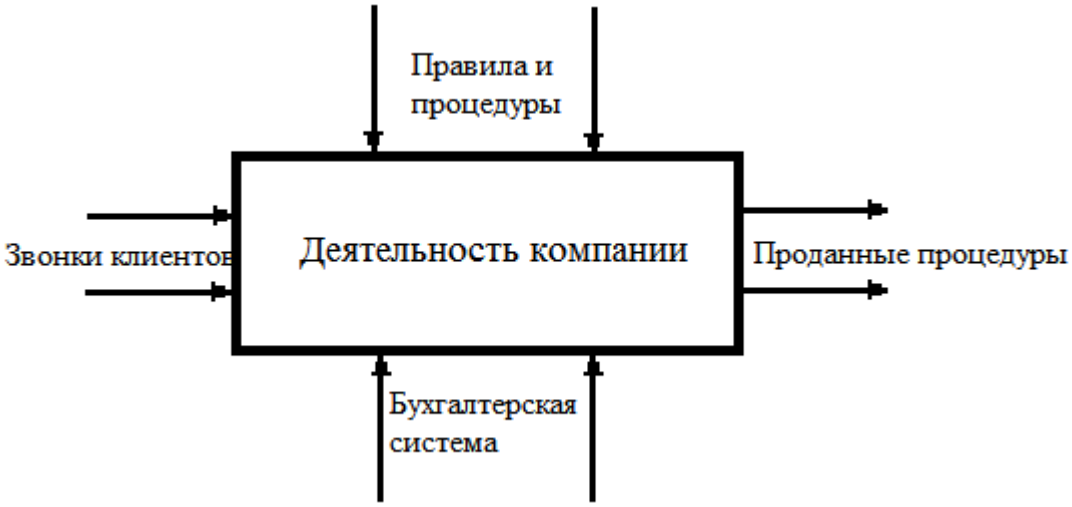


Рис. 2. Контекстная диаграмма (результат выполнения Упражнения 1)

Упражнение 2. Создание диаграммы декомпозиций

1. Выберите кнопку перехода на уровень ниже в панели инструментов.

2. В диалоговом окне укажите число работ на диаграмме нижнего уровня - "3", а нотацию декомпозиции - *IDEF* (рис. 3), затем нажмите "**ОК**". Автоматически будет создана диаграмма декомпозиции.

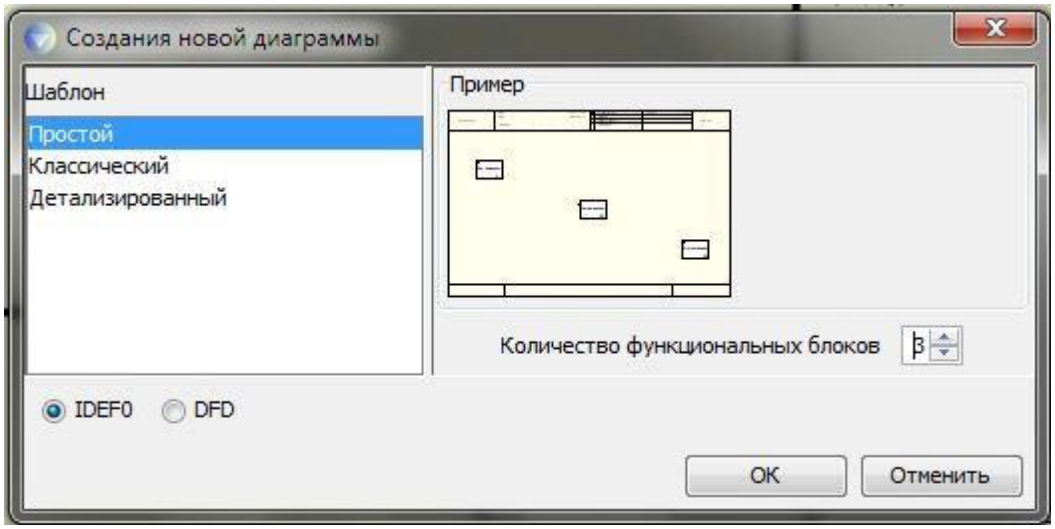


Рис. 3. Диалоговое окно декомпозиции работ.

3. Правой кнопкой мыши щелкните по 1-ой работе, выберите "**Редактировать активный элемент**" и на вкладке "**Название**" укажите имя работы. Повторите операцию для всех трех работ, а также внесите их описание в соответствующую вкладку на основе данных таблицы (табл. 2). Обратите внимание, что вкладка "**Описание**" может быть недоступна в версии RAMUS *Educational*.

Таблица 11.2. Описание работ декомпозиции первого уровня

Название	**Описание**
Продажа и маркетинг	Телемаркетинг, презентации, выставки
Сборка и тестирование компьютеров	Сборка и тестирование настольных и портативных компьютеров
Отгрузка и получение	Отгрузка заказов клиентам и получение компонентов от поставщиков

4. Перейдите в режим рисования стрелок. Произведите связывание граничных стрелок с функциональными объектами, как показано на рис. 4. Для связывания граничных стрелок наводите курсор на сами стрелки, а не на границы области построения моделей.

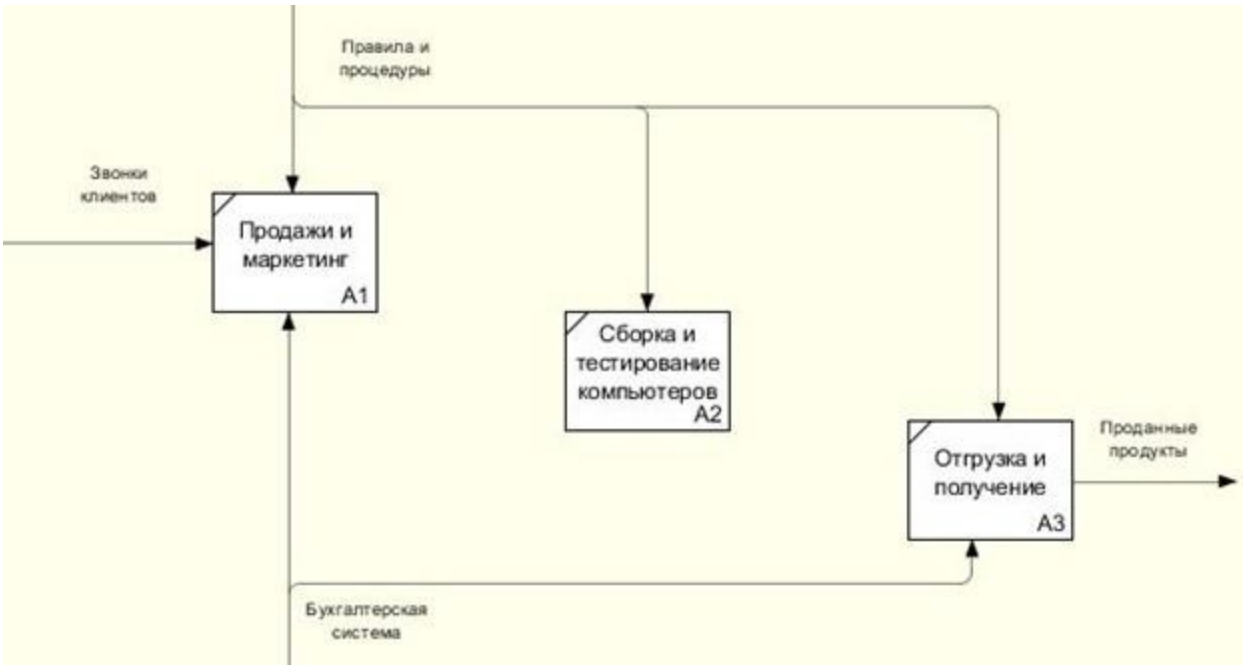


Рис. 4. Связывание граничных стрелок на диаграмме декомпозиции A0

Правой кнопкой мыши щелкните по ветви стрелки "**Сборка и тестирование компьютеров**", переименуйте ее в "**Правила сборки и тестирования**" (рис. 5).

5. Правой кнопкой мыши щелкните по ветви стрелки механизма работы "**Продажи и маркетинг**" и переименуйте ее в "**Система оформления заказов**" (рис. 5)

6. Создайте новые внутренние стрелки, как показано на рисунке (рис. 6)

7. Создайте стрелку обратной связи (по управлению) "**Результаты сборки и тестирования**", идущую от работы "**Сборка и тестирование компьютеров**" к "**Продажи и маркетинг**". Измените стиль стрелки - толщину (**правая кнопка мыши -> "Редактировать активный элемент" -> вкладка "Линия"**). Методом drag&drop возможно переносить стрелки и их названия. При необходимости возможно установить "тильду" (опция

контекстного меню при нажатии на стрелке правой кнопкой мыши) для явной связи стрелки и подписи к ней (рис. 7)

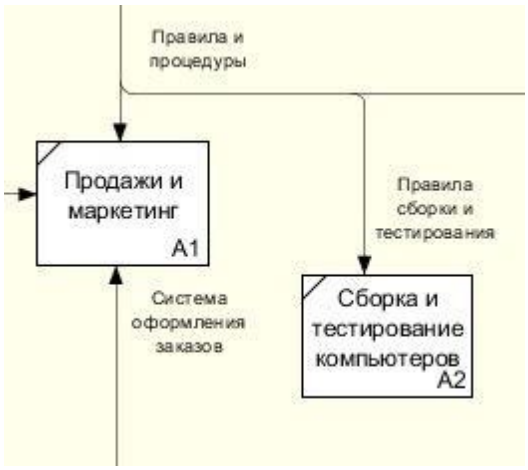


Рис. 5. Присвоение названий ветвям стрелок диаграммы декомпозиции A0.

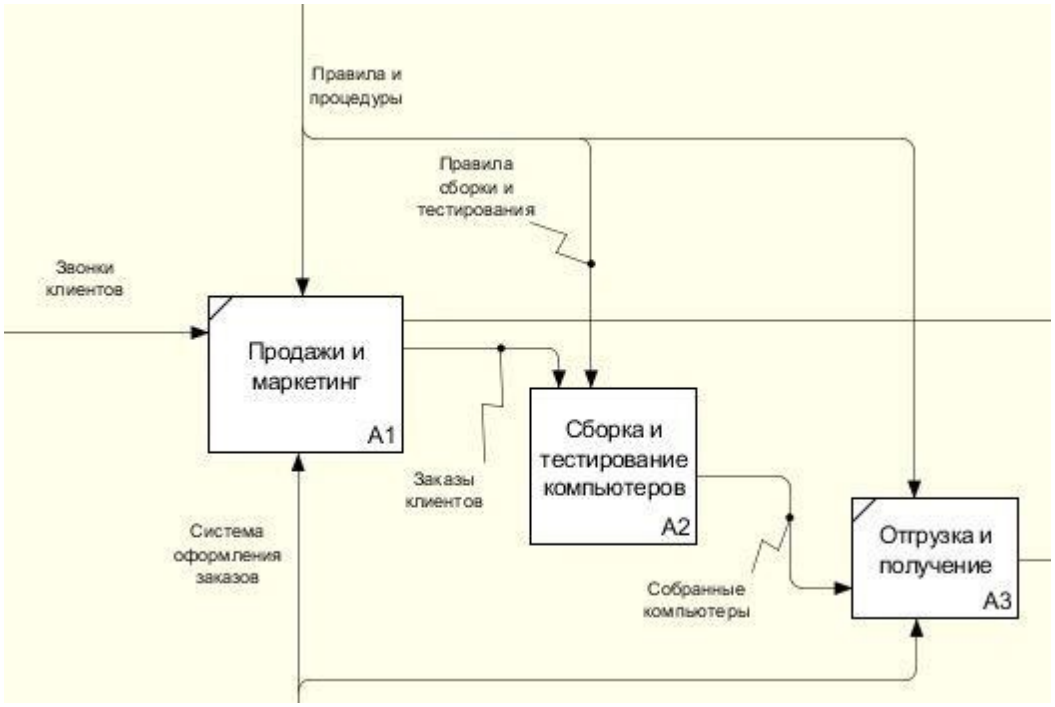


Рис. 6. Внутренние стрелки диаграммы декомпозиции A0.

8. Создайте новую граничную стрелку "**Маркетинговые материалы**", выходящую из работы "**Продажи и маркетинг**" Эта стрелка автоматически не попадает на диаграмму верхнего уровне и имеет квадратные скобки у окончания . Щелкните правой кнопки мыши по квадратным скобкам и выберите в контекстном меню "**Туннель**" (см. рис. 8) одну их двух опций: "**Создать стрелку**" и "**Обозначить туннель круглыми скобками**", в нашем случае - первый вариант.

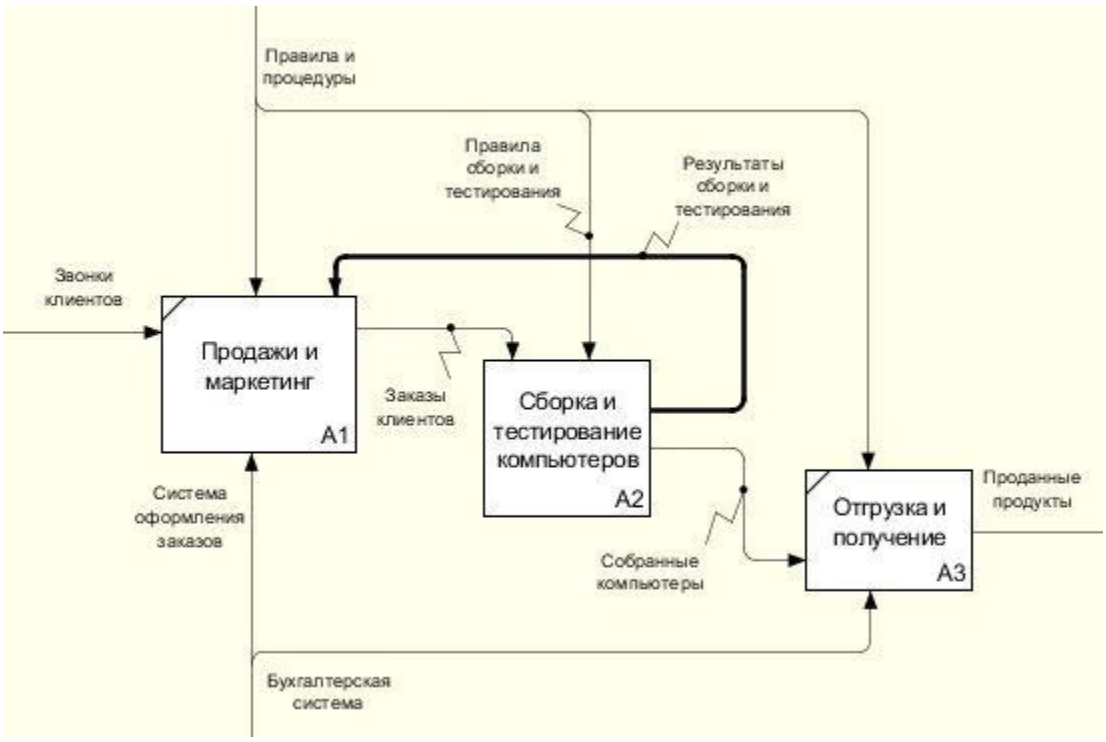


Рис. 7. Результаты редактирования стрелок на диаграмме декомпозиции A0.

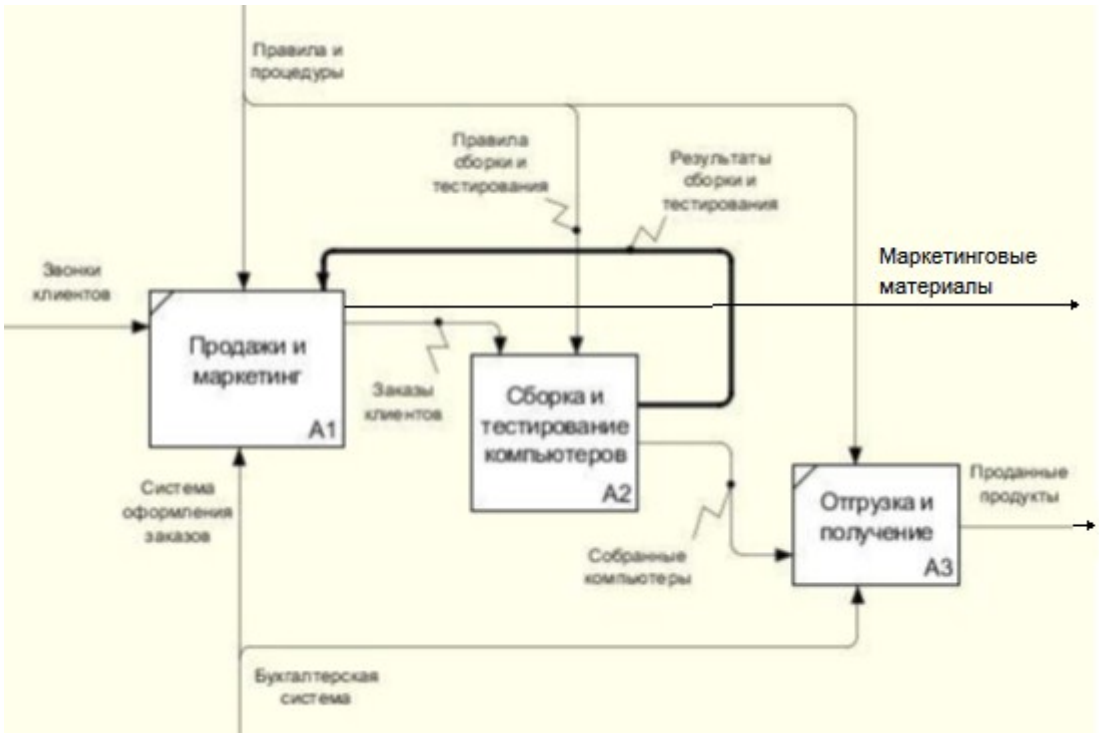


Рис. 8. Результат туннелирования стрелок

2. Метод функционального моделирования SADT. Методология IDEF0.

Цель работы: Изучить теоретические основы структурного подхода к проектированию информационных систем. Освоить принципы построения IDEF0-диаграммы классов в программной среде Ramus Educational.

Задачи:

1. Ознакомиться с теоретическими вопросами структурного подхода к проектированию информационных систем.

2. Изучить диаграмму IDEF0 (Integration Definition for Function Modeling) для предметной области «Гостиница».

3. Построить с помощью программного средства Ramus Educational диаграмму IDEF0 согласно индивидуальному заданию (вариант получить у преподавателя).

2.1 Теоретические сведения

2.1.1 Общие положения структурного метода

Сущность структурного подхода к разработке ИС заключается в декомпозиции (разбиении) системы на автоматизируемые функции, которые в свою очередь делятся на подфункции, на задачи и так далее. Процесс декомпозиции продолжается вплоть до определения конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимосвязаны.

В основе структурного метода лежит несколько общих принципов:

– разбиение системы на множество независимых задач, доступных для понимания и решения;

– иерархическое упорядочивание, т.е. организация составных частей проблемы в древовидные структуры с добавлением новых деталей на каждом уровне.

К основным принципам относятся:

– абстрагирование, т.е. выделение существенных аспектов системы и отвлечение от несущественных;

– формализация, т.е. общее методологическое решение проблемы;

– непротиворечивость, состоящая в обосновании и согласовании элементов системы;

– иерархическая структуризация данных.

2.1.2 Метод функционального моделирования SADT

На основе метода SADT, предложенного Д. Россом, разработана методология IDEF0 (Icam DEFinition), которая является основной частью программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой по инициативе ВВС США.

Методология IDEF0 является наиболее признанным эффективным средством анализа, конструирования и отображения бизнес-процессов, применяемым также и широко за пределами США.

Метод SADT применяется при моделировании широкого круга систем, для которых определяются требования и функции, после чего проводится их реализация.

Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели предметной области, которая отображает функциональную структуру, производимые функции и действия, а также связи между ними.

Результат применения метода SADT - модель, которая состоит из диаграмм, фрагментов текстов и глоссария со ссылками друг на друга. Все функции и интерфейсы представляются диаграммами в виде, соответственно, блоков и дуг. Место соединения дуги с блоком определяет тип интерфейса.

Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке (исходные данные), указывается с левой стороны блока, а результаты работы функции (выход, результат) - с правой стороны. Механизм, осуществляющий операцию (человек или автоматизированная система), задается дугой, входящей в блок снизу (см. рис. 1).

Описание системы с помощью SADT называется моделью. Субъектом моделирования служит сама система. Однако моделируемая система никогда не существует изолированно: она всегда связана с окружающей средой. По этой причине в методологии SADT подчеркивается необходимость точногоопределения границ системы, т.е. модель устанавливает точно, что является и что не является субъектом моделирования, описывая то, что

входит в систему, и, подразумевая то, что лежит за ее пределами. SADT-модель должна иметь единственный субъект.

С определением модели тесно связана позиция (называемая точкой зрения), с которой наблюдается система и создается ее модель. "Точку зрения" лучше всего представлять себе как место (роль, должность) человека или объекта в рассматриваемой системе, на которое надо «встать», чтобы увидеть систему в действии и необходимой полноте. У конкретной модели может быть только **одна** точка зрения.

Обычно вопросы для SADT-модели формулируются на самом раннем этапе проектирования, при этом основная суть этих вопросов должна быть выражена в одной-двух фразах, которые становятся **целью** модели.

После того как определены субъект, цель и точка зрения модели, начинается первая интеграция процесса моделирования по методологии SADT. Субъект определяет, что включить в модель, а что исключить из нее.

Точка зрения диктует автору модели выбор нужной информации о субъекте и форму ее представления. Цель становится критерием окончания моделирования. Конечным результатом этого процесса является набор тщательно взаимоувязанных описаний, начиная с описания самого верхнего уровня системы и заканчивая подробным описанием ее деталей или отдельных операций.

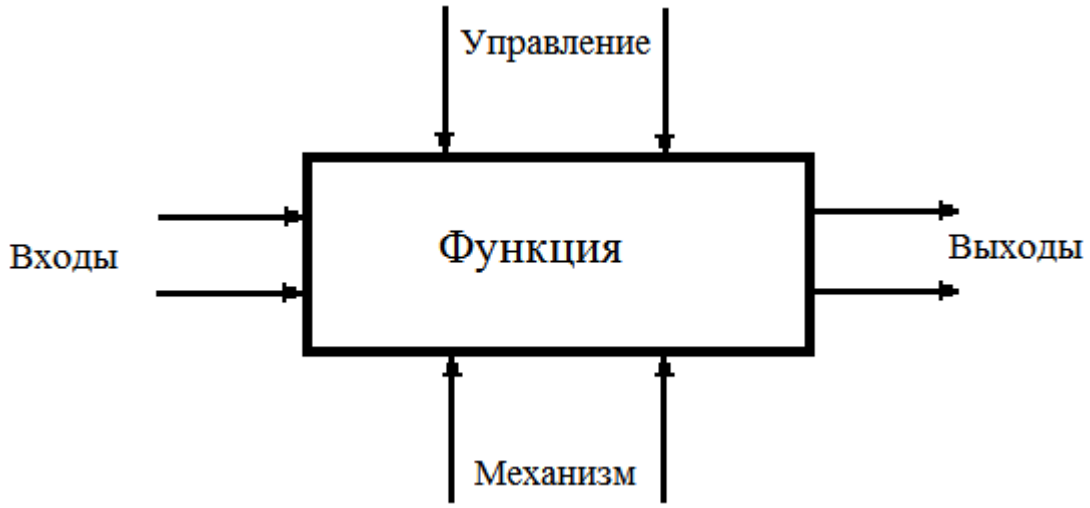


Рис. 1. Структура модели

Каждое из таких тщательно взаимосогласованных описаний называется диаграммой и имеет определенный уровень детализации. SADT-модель объединяет и организует диаграммы в иерархические структуры, в которых диаграммы наверху модели менее детализированы, чем диаграммы нижних уровней. Другими словами, модель SADT можно представить в виде древовидной структуры диаграмм, где верхняя диаграмма является наиболее общей, а самые нижние – максимально детализированы (см. Рис. 2).

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть детализирован диаграммой нижнего уровня, которая, в свою очередь, также может детализироваться с помощью необходимого числа диаграмм. Таким образом, формируется иерархия диаграмм. Для того чтобы указать положение любой диаграммы или блока в иерархии, им присваивают уникальные обозначения. Например, **A41** (**A** сокр. от **Activity**) является диаграммой, которая детализирует блок 1 на диаграмме **A4**. Аналогично, **A4** детализирует блок 4 на диаграмме **A0**, которая является самой верхней (родительской) диаграммой модели.

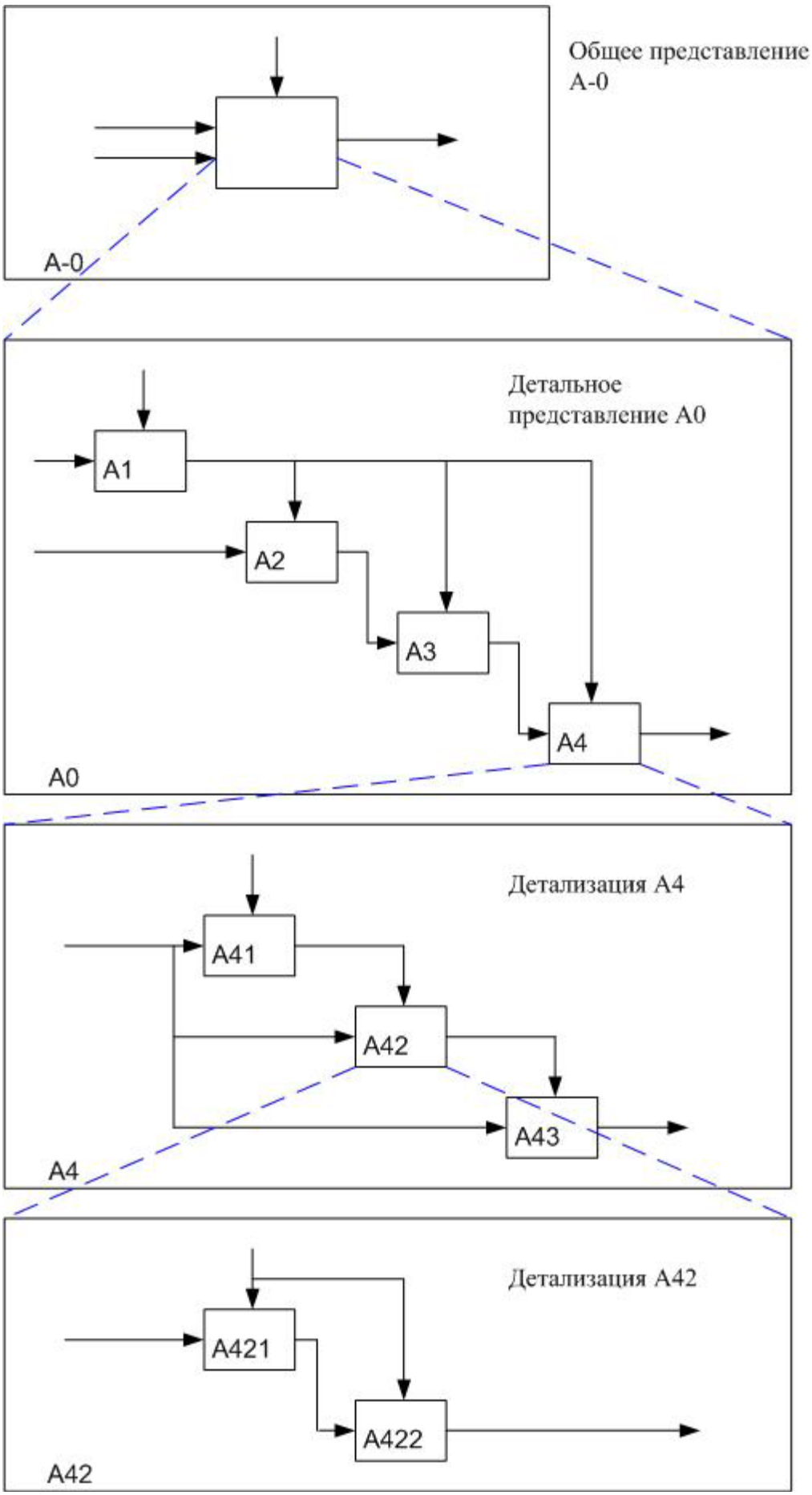


Рис. 2. Структура SADT-модели. Иерархия и декомпозиция диаграмм

Некоторые дуги имеют начало в одном из блоков диаграммы и завершение в другом, у других же начало может исходить от границ диаграммы – дуги управления, механизма, дуги входа и выхода, перенесенные с родительской (верхнего уровня) диаграммы.

Таким образом, источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме.

Также следует сказать о так называемых «туннельных дугах».

Туннельные дуги означают, что данные, выраженные этими дугами не рассматриваются на следующем уровне детализации (как бы проходят «насквозь»). Если «туннель» расположен в месте соединения дуги с блоком, то данные этой дуги не обязательны на следующем уровне детализации. Если же «туннель» находится на противоположном конце дуги - это значит, что данные дуги не описываются на родительской диаграмме. Граничные дуги должны продолжаться (дублироваться) на родительской диаграмме, делая ее полной и непротиворечивой.

Для упрощения понимания приведенных диаграмм, следует расшифровать применяемую в IDEF систему обозначений, позволяющую аналитику точно идентифицировать и проверять по дугам связи между диаграммами. Эта схема кодирования дуг - "**ICOM**" - получила название по первым буквам английских эквивалентов слов вход (**Input**), управление (**Control**), выход (**Output**), механизм (**Mechanism**).

Программное обеспечение «Ramus» предназначено для использования в проектах, в которых необходимо описание бизнес-процессов предприятия. «Ramus» поддерживает методологии моделирования бизнес-процессов IDEF0 и DFD, а также имеет ряд дополнительных возможностей, призванных удовлетворить потребности команд разработчиков систем управления предприятиями. «Ramus» обладает гибкими возможностями построения отчетности по графическим моделям, позволяющие создавать отчеты в форме документов, регламентирующих деятельность предприятия.

Ramus Educational имеет достаточно интуитивный интерфейс пользователя, позволяющий быстро и просто создавать сложные модели.

Начало работы

1. Запустите программу Ramus Educational. В появившемся окне (см. рис. 3) предлагается создать новый проект или открыть уже существующий.

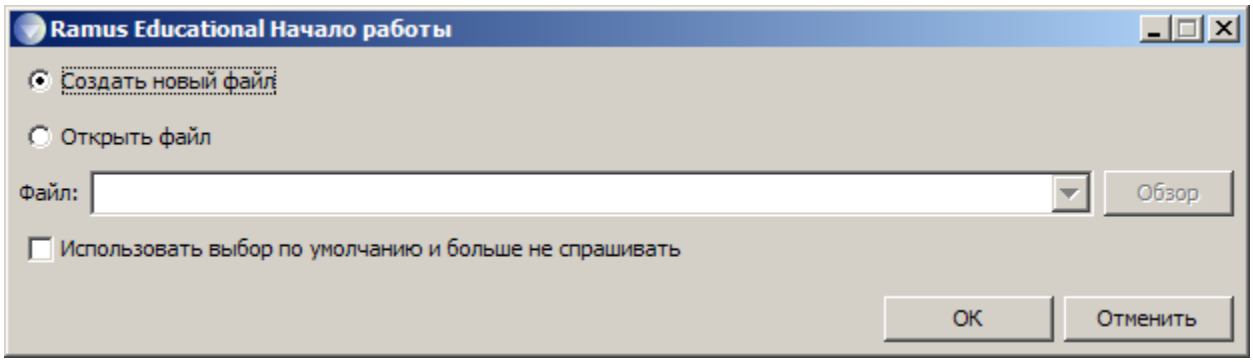


Рис. 3. Окно запуска Ramus Educational

2. После нажатия на кнопку «ОК» осуществляется запуск мастера проекта.

– на первом шаге (Рис. 4) в соответствующие поля необходимо внести сведения об авторе, названии проекта и модели, а также выбрать тип нотации модели (IDEF0 или DFD).

– на втором шаге вводится название организации, использующей данный проект.

– на третьем – дается краткое описание будущего проекта.

– четвертый шаг позволяет создать несколько основных классификаторов (в данном случае можно пропустить этот шаг). Так как модели процессов реальных предприятий могут содержать значительное количество объектов (документы, персонал, функции и т.д.), то в Ramus предусмотрена возможность упорядочено хранить информацию об этих объектах в виде системы классификаторов.

Классификация объектов упрощает поиск и обработку информации об объектах модели, а так же и об объектах непосредственно не представленных на диаграммах процессов, но относящихся к процессам предприятия.

– на пятом, заключительном, предлагается выбрать те из созданных классификаторов, элементы которых будут содержаться в перечне собственников процессов (пропустить данный шаг).

При необходимости можно завершить работу мастера, нажав кнопку «**Окончить**».

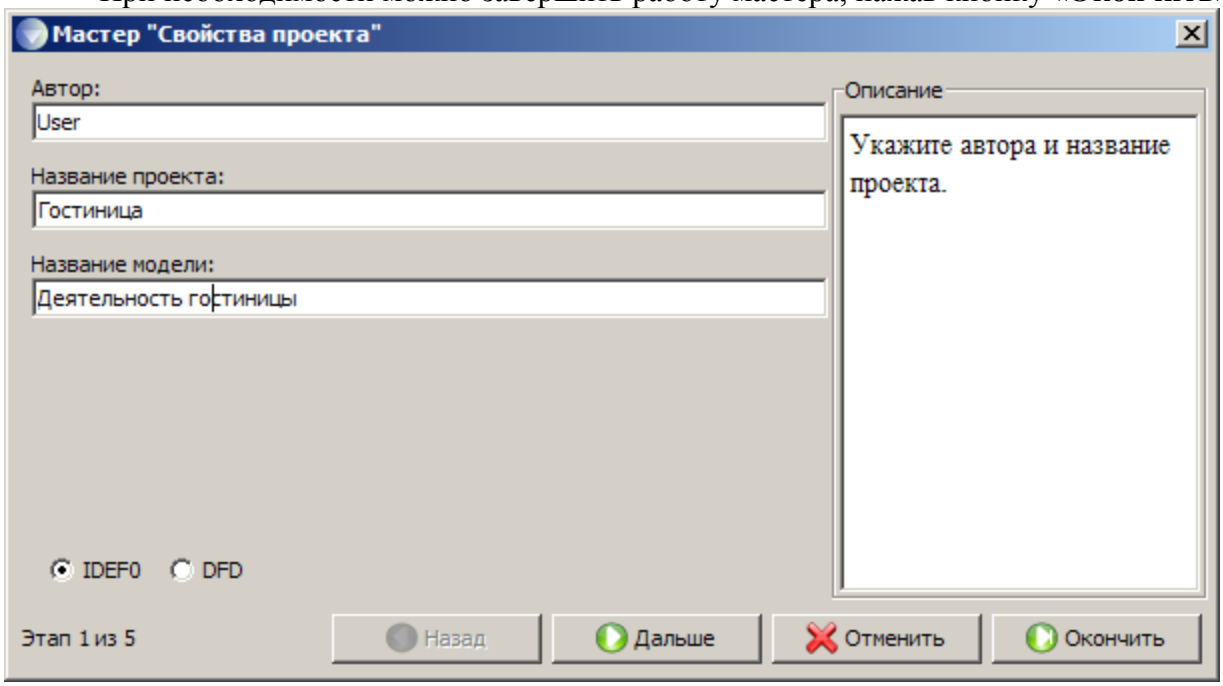


Рис. 4. Окно мастера создания проекта. Шаг 1

После завершения работы мастера, откроется рабочее пространство «Диаграммы», в котором можно приступить к рисованию графической модели (рис. 5). В верхней части приводятся сведения о проекте, введенные пользователем посредством мастера диаграмм.

Программа Ramus Educational обладает гибким графическим интерфейсом, который можно настроить под нужды и предпочтения конкретного пользователя: ненужные окна можно закрыть/свернуть; можно менять их размеры и месторасположение; также можно группировать два и более окон в одном, при этом содержимое вложенных окон будет размещено на вкладках общего окна (данный функционал возможен не для всех комбинаций окон).

3. Сохраните созданную модель, выбрав опцию меню «Файл» «Сохранить как».

Создание контекстной диаграммы

1. На панели инструментов выберите пиктограмму функции и мышью укажите месторасположение на рабочем пространстве.

2. Дайте данному функциональному блоку имя «**Описать деятельность гостиницы**».

3. Используя пиктограмму панели инструментов, создайте стрелки на контекстной диаграмме согласно Таблица 1.

Таблица 1. Контекстная диаграмма

Наименование стрелки	Тип
Заявки на заселение в гостиницу	вход
Материально-техническое и кадровое обеспечение работы гостиницы	вход
Нормативно-правовая база	управление
Гостиничные услуги	выход
Отчетность	выход
Финансовые результаты деятельности	выход
Услуги контрагентов	механизм
Инфраструктура гостиницы	механизм

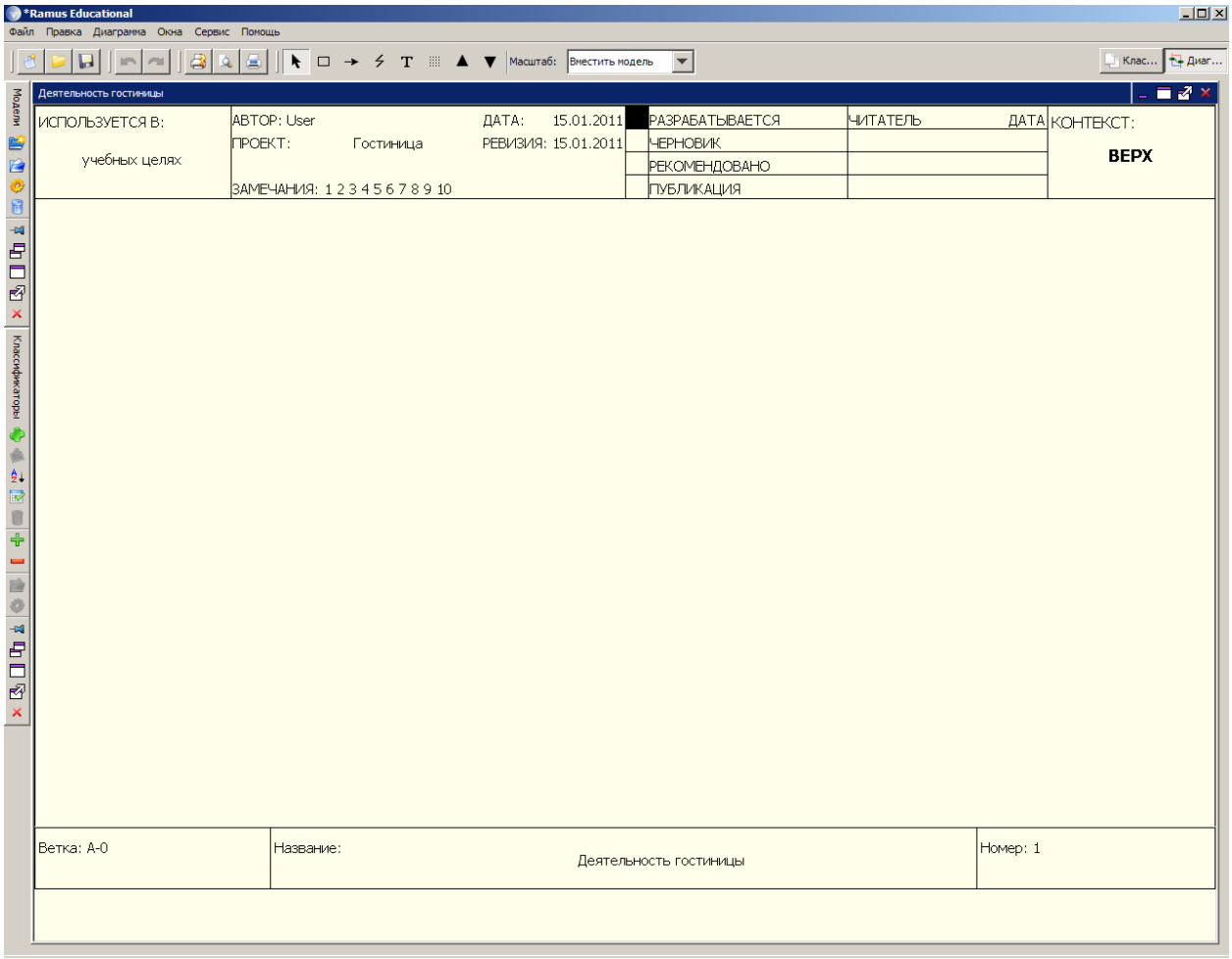


Рис. 5. Рабочее пространство «Диаграммы»

3. В результате должна получиться контекстная диаграмма, показанная на рис. 6.

Создание диаграммы декомпозиции

1. Выберите в палитре инструментов кнопку перехода на нижний уровень, в диалоговом окне «**Создание новой диаграммы**» (рис. 7) установите количество функциональных блоков **3**, укажите тип диаграммы (**IDEF0**) и нажмите кнопку **ОК**.

2. Автоматически будет создана диаграмма первого уровня декомпозиции (см. рис. 8) с перенесенными в нее потоками родительской диаграммы.

3. Осуществите построение детализирующих диаграмм для функциональных блоков согласно представленной иерархии (см. рис. 9).

4. В результате должны быть разработаны IDEF0-диаграммы, представленные на рис. 11 - рис. 15.

5. В программе Ramus Educational предусмотрена возможность экспорта разработанных диаграмм в виде рисунков формата *.png, *.bmp или *.jpeg. Для этого в главном меню необходимо выбрать команду «**Диаграммы**» «**Экспортировать как рисунки**». В появившемся окне указывается список экспортируемых рисунков, выбирается их формат и размер, а также путь для сохранения (см. рис. 16).

6. Покажите преподавателю разработанные диаграммы.

Содержание работы

1. Ознакомиться с теоретическими вопросами построения IDEF0-диаграммы.

2. Изучить диаграмму IDEF0 для предметной области «Гостиница».

3. Осуществить построение IDEF0-диаграммы с помощью программного средства Ramus Educational согласно индивидуальному заданию (вариант получить у преподавателя).

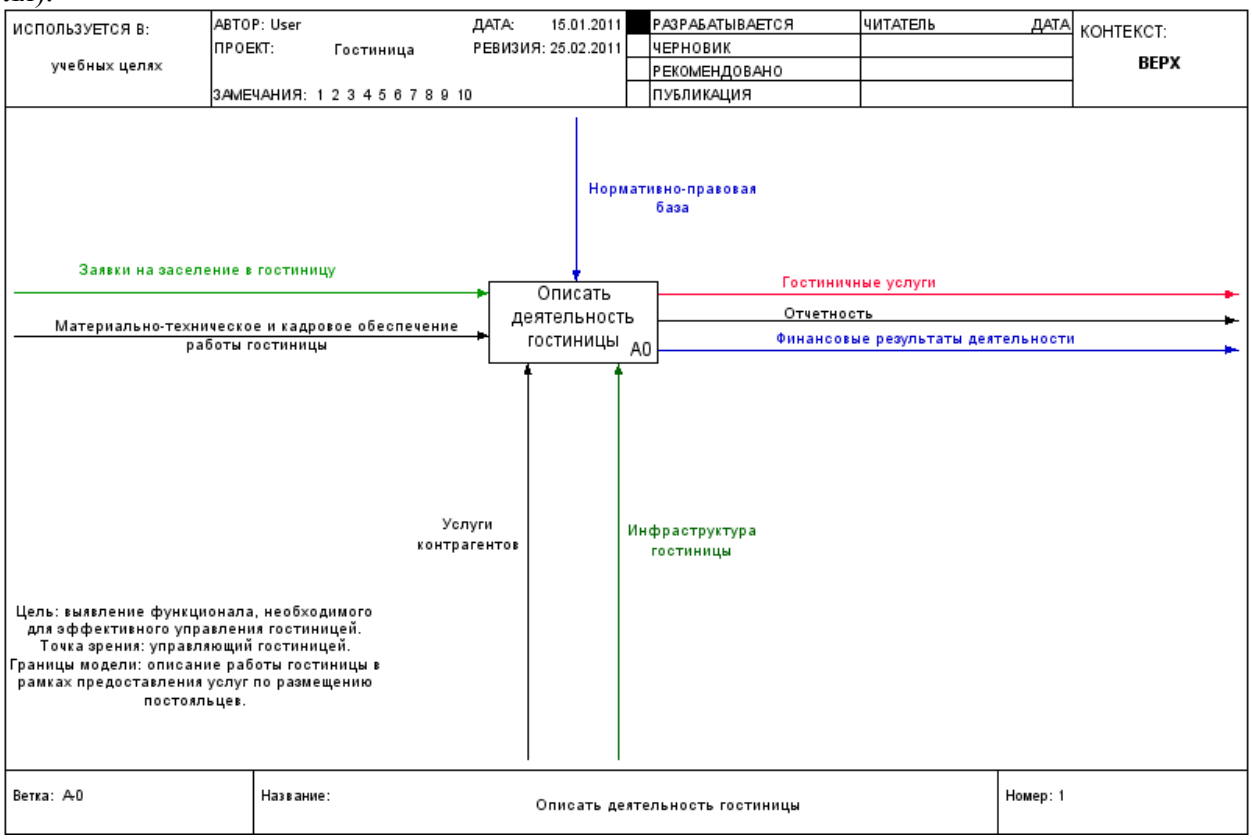


Рис. 6. Контекстная диаграмма предметной области «Гостиница»

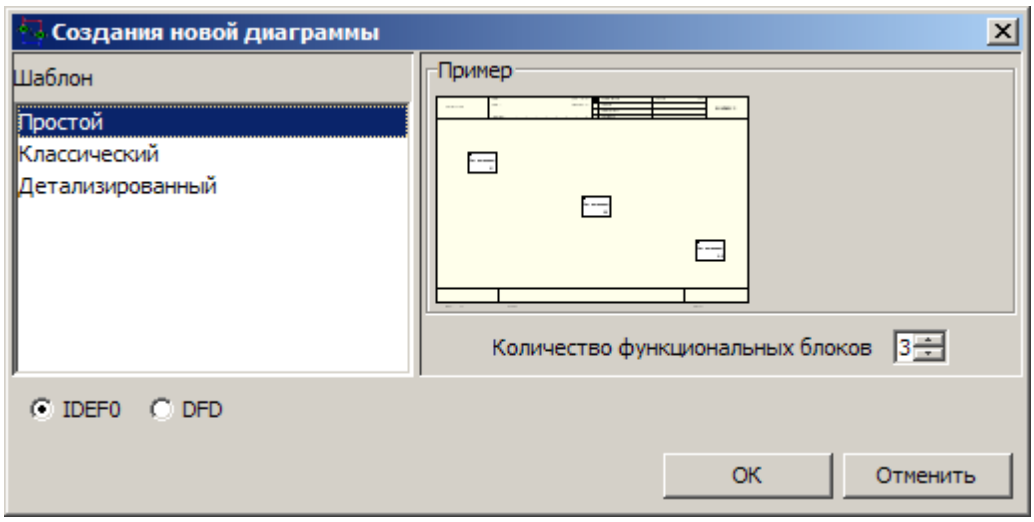


Рис. 7. Диалоговое окно создания детализирующей диаграммы.

Требования к отчету

Отчет по лабораторной работе оформляется в печатном виде. Защита работы включает в себя проверку знания студентом теоретического материала, а также практической части лабораторной работы.

Отчет должен включать:

- Разработанные IDEF0-диаграммы.
- Описание разработанных IDEF0-диаграмм.

Все примеры должны быть сохранены на сетевом диске или на диске студента.

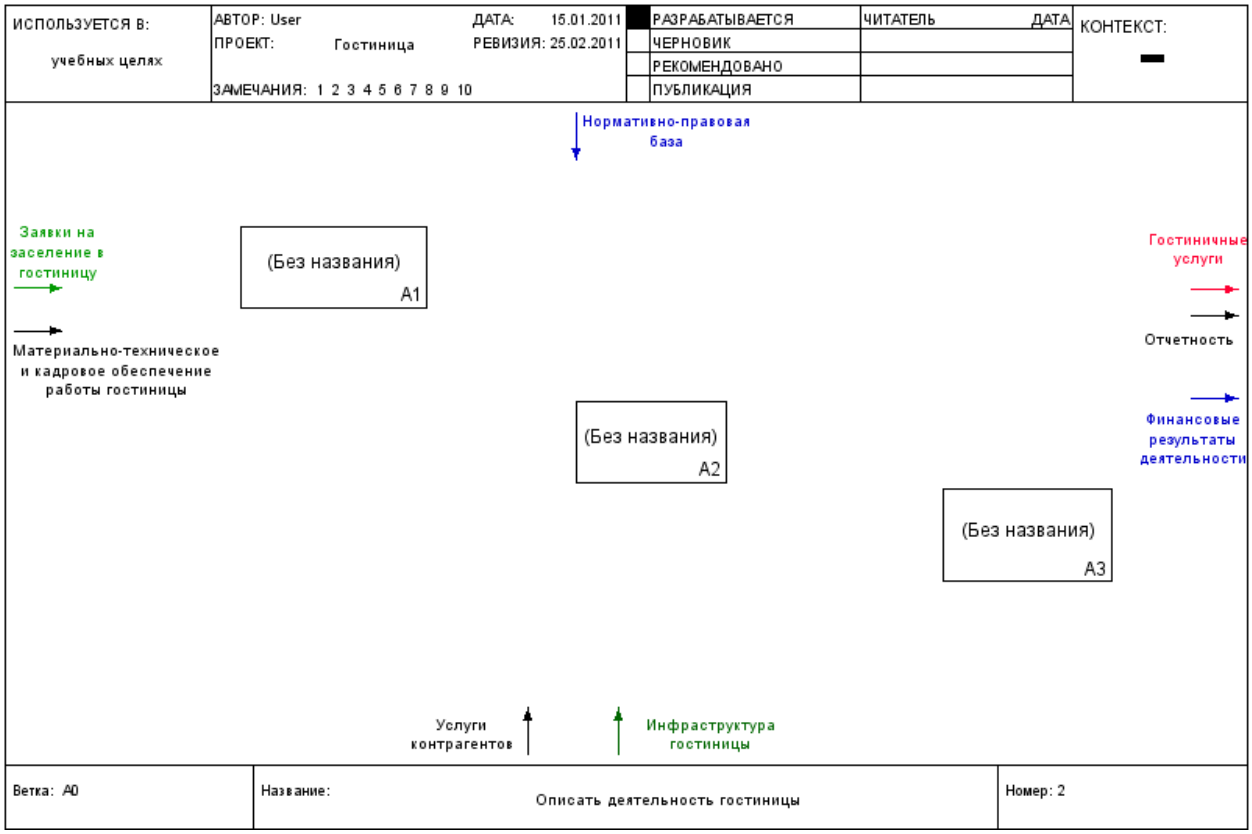


Рис. 8. Рабочее пространство детализирующей диаграммы.

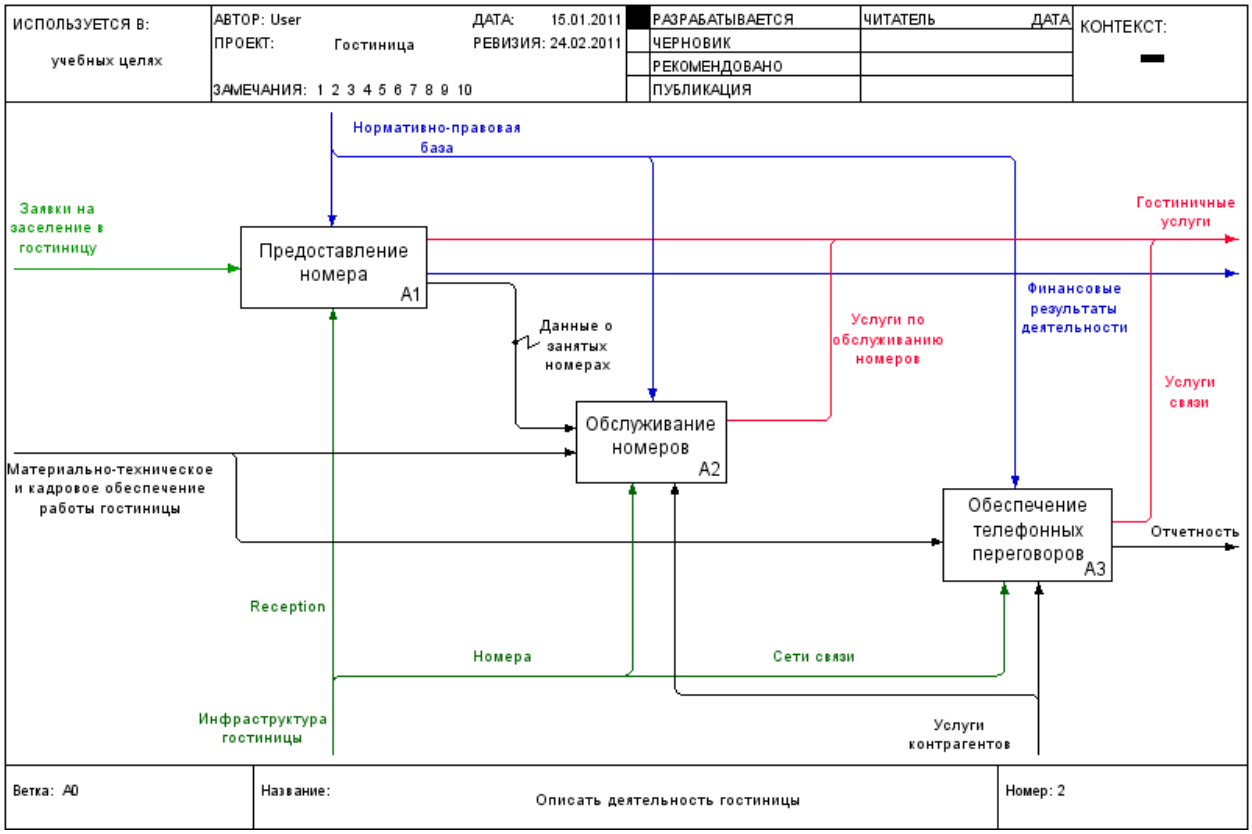


Рис. 9. Детализированная диаграмма первого уровня.

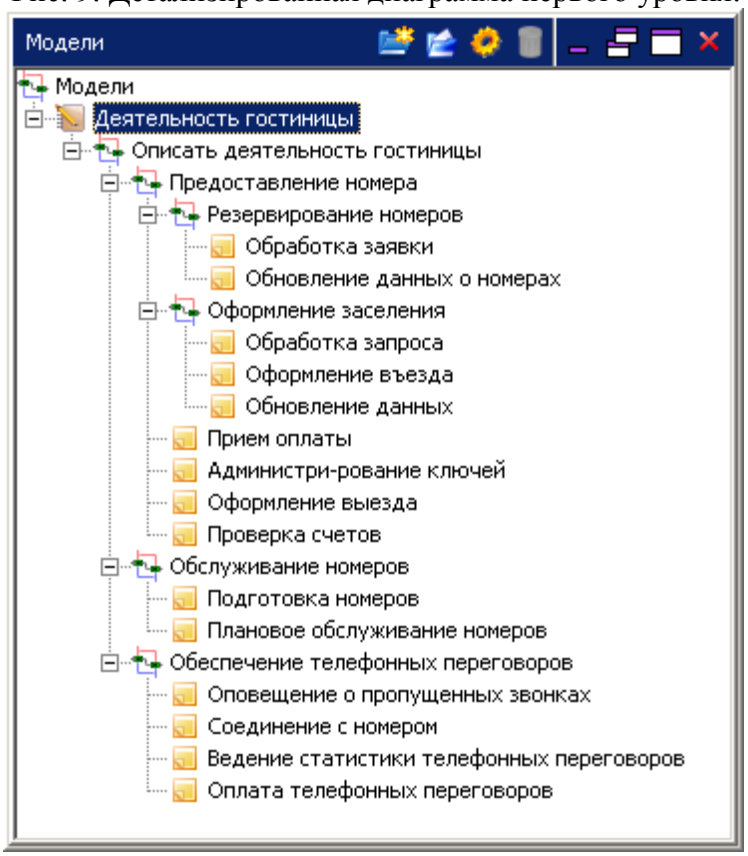


Рис. 10. Иерархия функциональных блоков модели

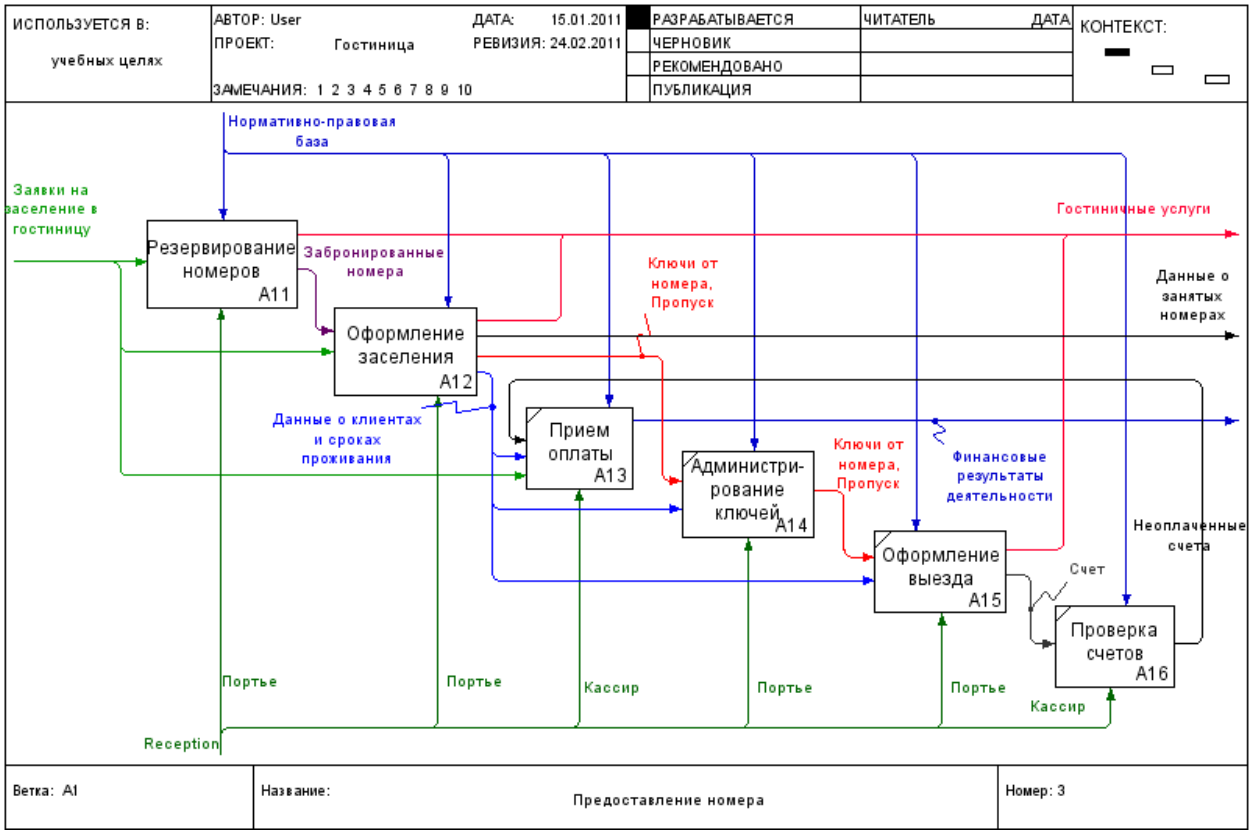


Рис. 11. Детализированная диаграмма работы «Предоставление номера».

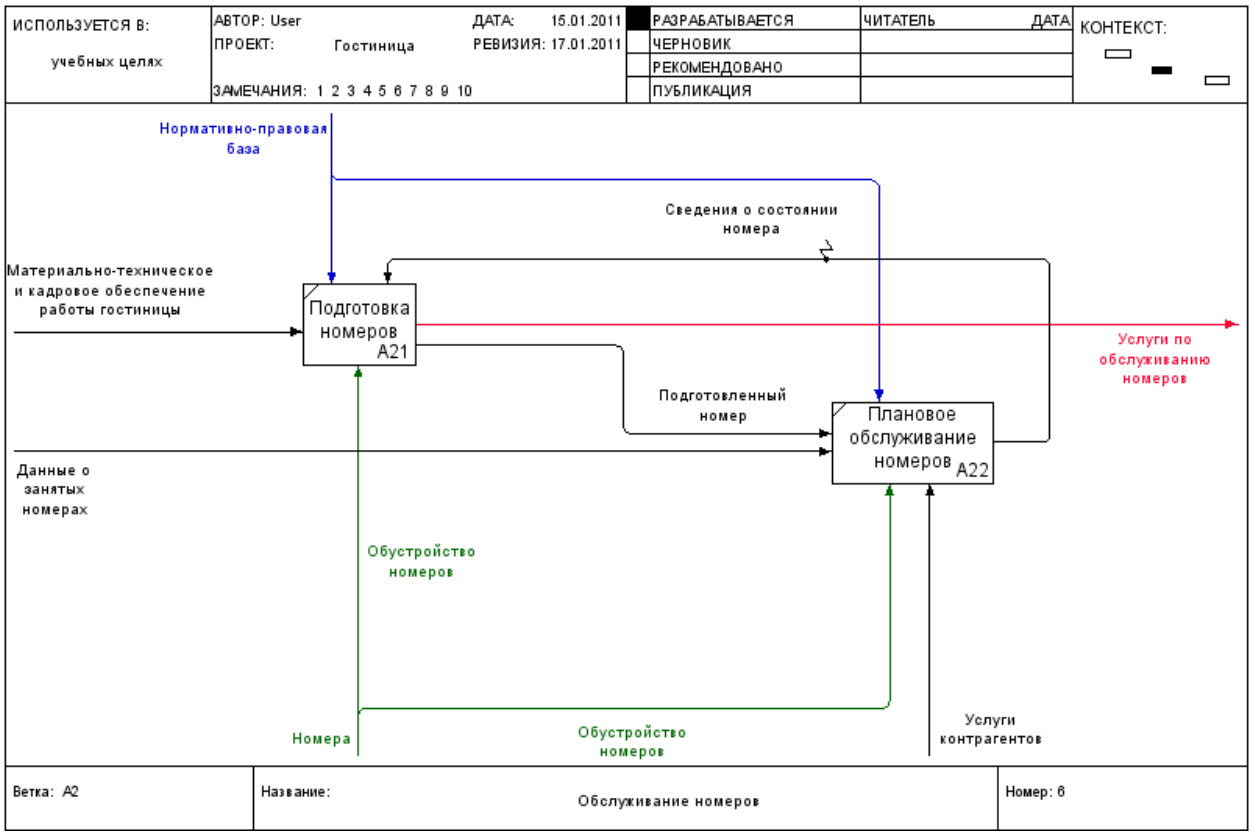


Рис. 12. Детализированная диаграмма работы «Обслуживание номеров».

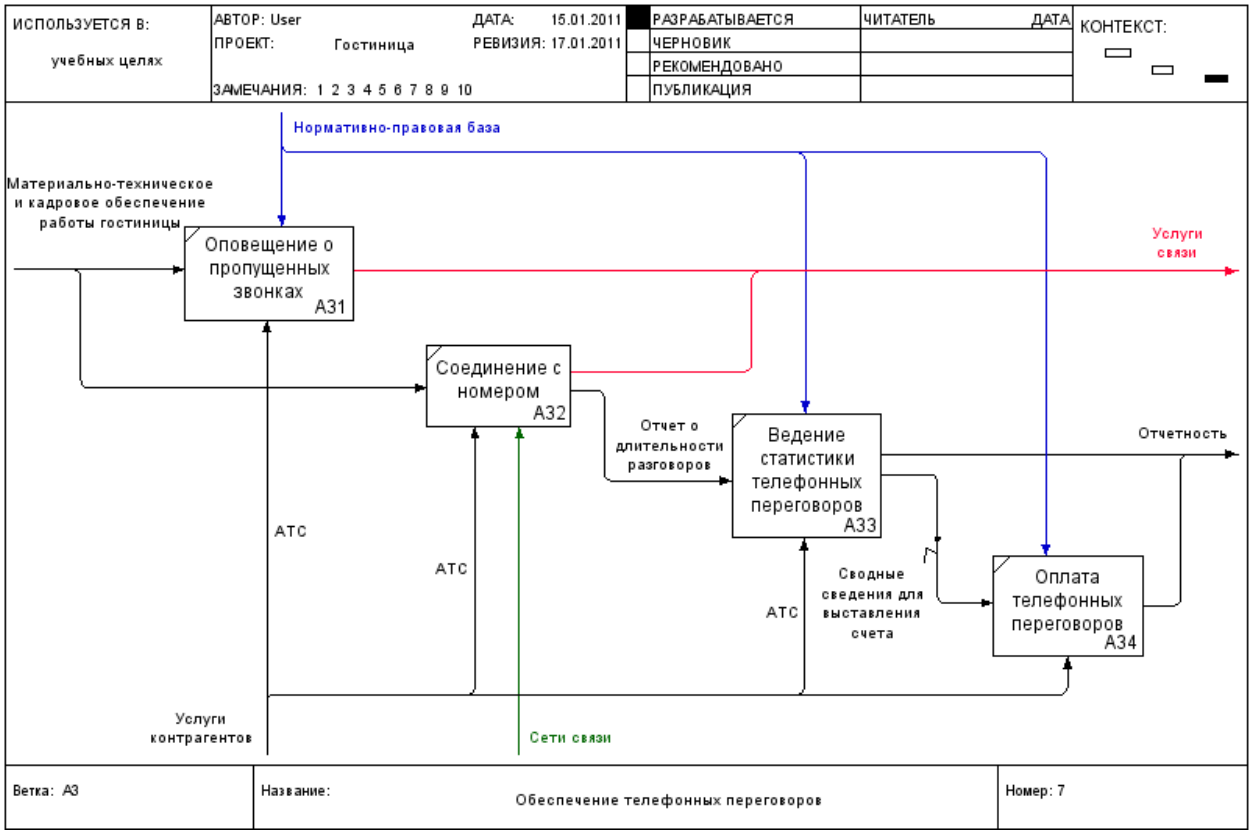


Рис. 13. Детализированная диаграмма работы «Обеспечение телефонных переговоров».

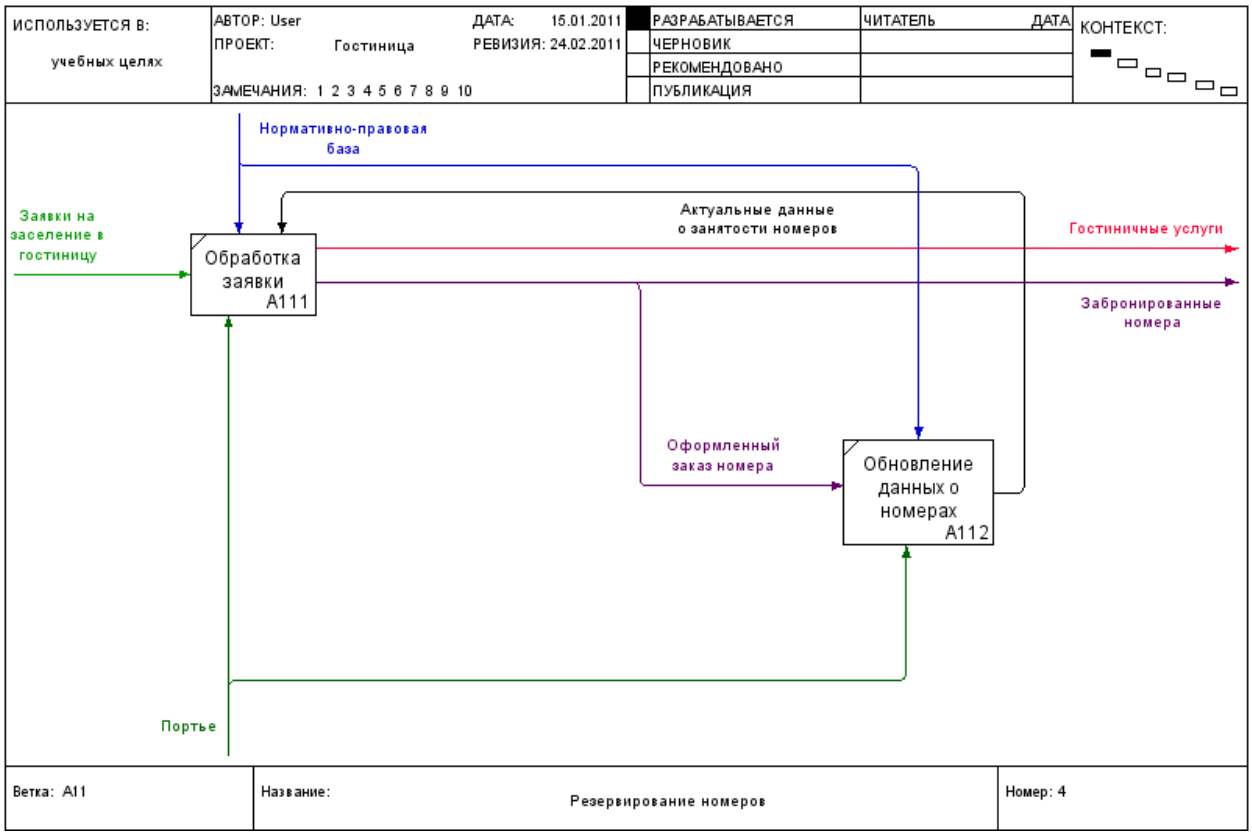


Рис. 14. Детализированная диаграмма работы «Резервирование номеров».

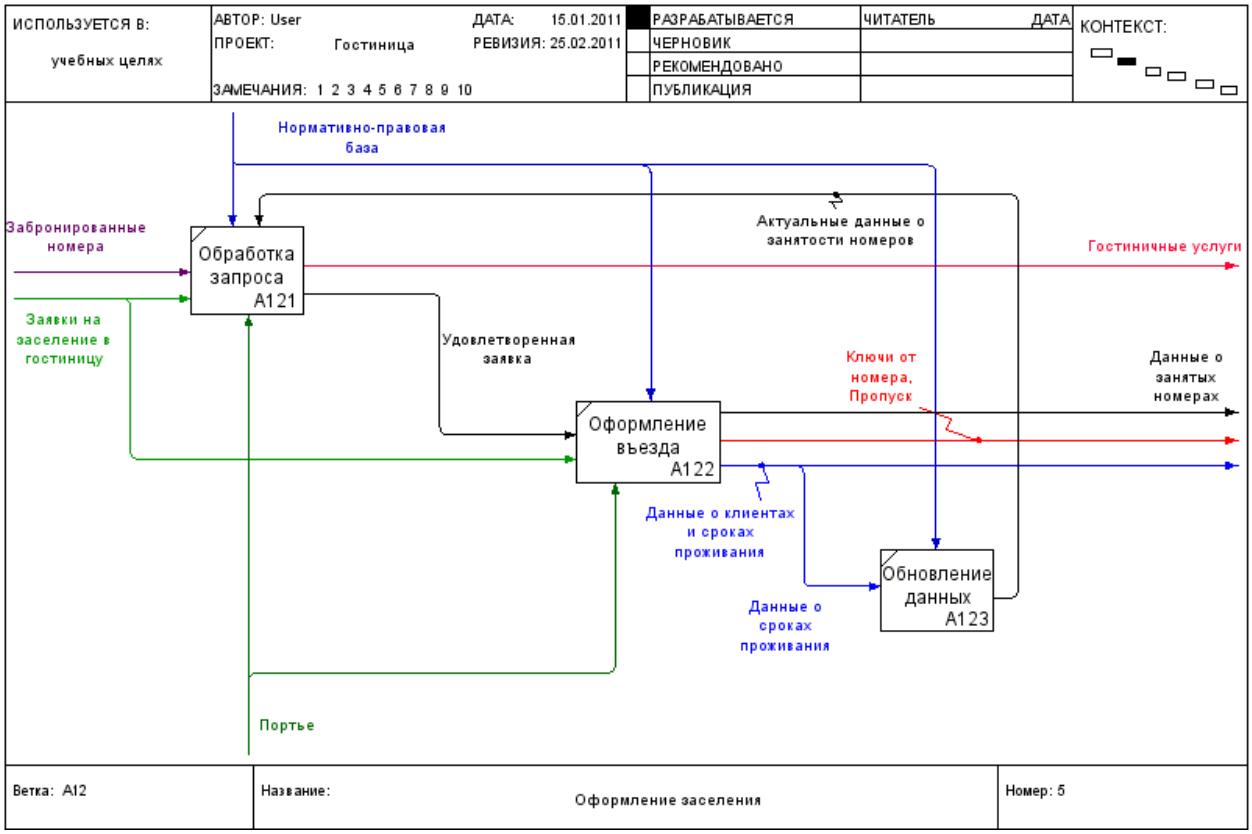


Рис. 15. Детализированная диаграмма работы «Оформление заселения»

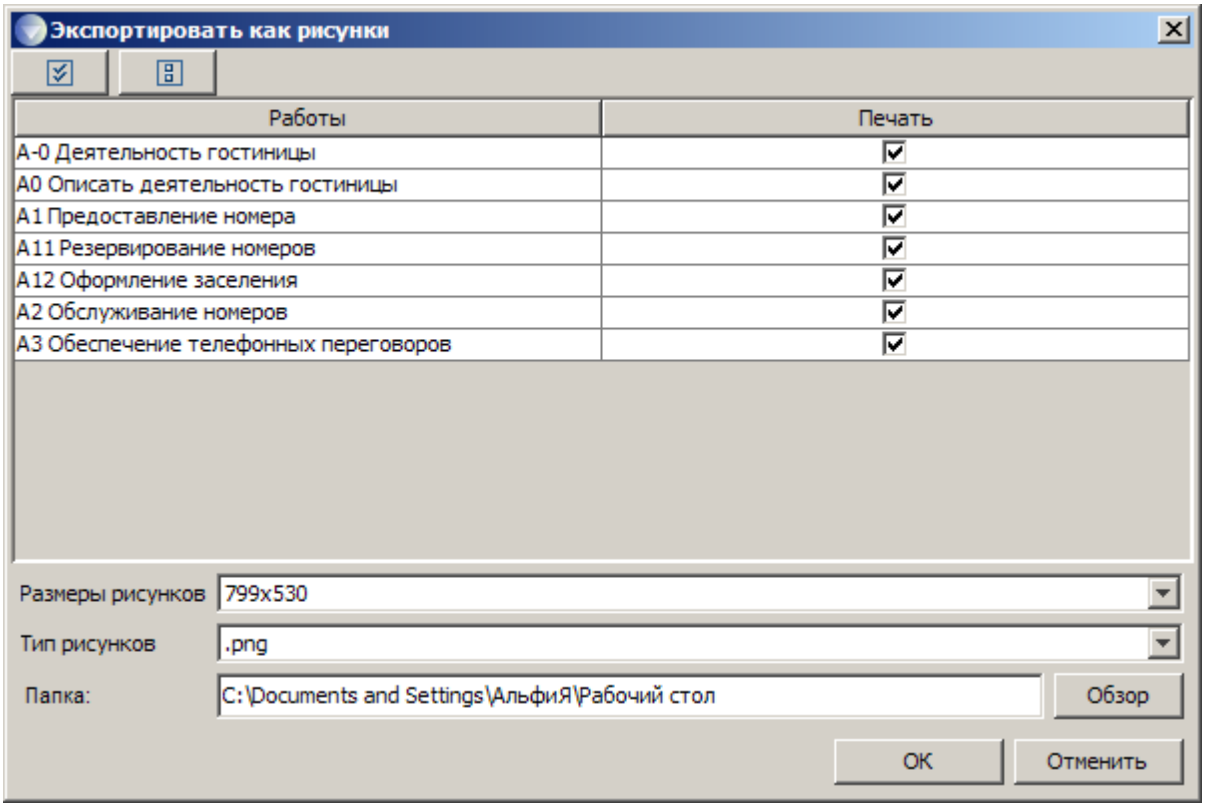


Рис. 16. Диалоговое окно экспорта диаграмм.

ВАРИАНТЫ ЗАДАНИЙ

Вариант задания выбирается по сумме двух последних цифр зачетной книжки.

1. Автострахование.
2. Агентство по сдаче автомобилей в аренду.
3. Аренда коньков, роликов, велосипедов, лыж.
4. Аэропорт – пассажирское расписание и перевозки.
5. Банковская система вкладов (физических и юридических лиц) .
6. Банковская система кредитования (физических и юридических лиц).
7. Биллинг сотовой компании.
8. Ветеринарная лечебница.
9. Клуб обучения танцам.
10. Магазин косметики.
11. Машиностроительное предприятие: система по разработке и модификации изделий (ведение архива, стандартов и пр.).
12. Нефтеперерабатывающая компания.
13. Парикмахерская.
14. Поставка вин.
15. Приемная комиссия ВУЗа.
16. Производство мебели (прием индивидуальных и типовых заказов и изготовление).
17. Рекламное агентство.
18. Риэлторская компания: аренда; продажа первичного и вторичного жилья.
19. Санаторий.
20. Система управления проектом для IT-компании.
21. Складская логистика.
22. Спа-салон (услуги, обслуживающий персонал и пр.).
23. Страховая компания.
24. Такси.
25. Транспортная логистика.
26. Туристическое агентство (путешествия за рубеж).
27. Туристическое агентство (путешествия по России).
28. Учет оборудования на крупном промышленном предприятии.
29. Филармония.
30. Электронный проездной.

3. Решение поставленной задачи при помощи технологии IDEF0.

Цель работы: получение навыков создания и редактирования функциональных моделей в Ramus Educational. Для выполнения последующей лабораторной работы необходимо иметь результат выполнения предыдущей, поэтому рекомендуется сохранять модель, полученную в конце каждого упражнения.

Задание: требуется разработать модель программного обеспечения информационной системы, обеспечивающей возможность автоматизированного управления клиентом образовательного интернет сайта.

Описание системы:

Интернет сайт должен использоваться для дистанционной поддержки обучения студентов кафедры ИС. На сайте должны быть размещены материалы, необходимые для изучения студентами дисциплин, преподаваемых на кафедре, преподаватели, ведущие занятия со студентами, расписание занятий, программное обеспечение, распространяемое бесплатно и необходимое студентами для изучения материалов учебных дисциплин, а так

же новости и объявления, публикуемые преподавателями кафедры. На сайте должны быть представлены тесты, предназначенные для проверки знаний студентов по каждой дисциплине. Тесты доступны только зарегистрированным студентам. Результаты теста фиксируются в системе, а преподаватель получает уведомление о результатах прохождения тестов студентами. На сайте должен быть реализован режим анкетирования, при котором студенту предлагается заполнить анкету, т.е. дать ответы на некоторые вопросы, предложенные преподавателями. Анкетирование предназначено для выявления недостатков преподавания и корректировки системы обучения на кафедре.

Работу с сайтом можно у словно разделить на работу студента, работу преподавателя, работу администратора. Студент, обратившись к сайту, имеет возможность просматривать все его разделы, скачивать файлы и т.д.. по не имеет возможности проходить тестирование и участвовать в анкетировании. Для того, чтобы эти возможности стали доступны студенту, ему необходимо пройти процесс регистрации, в ходе которого он выбирает себе НИК (login) и пароль, а так же вводит необходимые сведения о себе. В последствии, при подключению к сайту, студент должен иметь возможность авторизоваться на нем. введя свой ник и пароль для получения полного доступа ко всем ресурсам сайта. При прохождении теста, его результаты запоминаются в базе данных. Таким образом, накапливается статистика о результатах обучения студента, которую может использовать преподаватель для выставления окончательной экзаменационной оценки студенту по данной дисциплине. Преподаватель, обратившись к сайту имеет те же возможности, что и студент до тех пор . пока не пройдет процедуру авторизации (вход в систему). Для осуществления возможности входа в систему преподаватель так же должен пройти процедуру регистрации в системе, в ходе которой ввести ник. пароль и дополнительную информацию о себе. После входа в систему, преподаватель, помимо просмотра всех разделов сайта, ответов на контрольные вопросы и анкетирования, имеет возможность публиковать новости и объявления в соответствующих разделах сайта, а так же размещать все возможные материалы, касающихся своих курсов и других видов учебной работы. Кроме прочего, преподавателю должна быть доступна статистика ответов на вопросы контрольных тестов и анкет всех дисциплин, публикуемых на сайте. Преподаватель так же имеет возможность просматривать и корректировать списку студентов, обучающихся в «его» группах, а так же публиковать в специальном разделе сайта рейтинговые и экзаменационные оценки.

Администратор, помимо обеспечения функционирования сайта, осуществляет поддержку и контроль его информационной и программной целостности и работоспособности. Администратор имеет доступ ко всем разделам сайта, осуществляет поддержку и контроль ею информационной и программной целостности и работоспособности. Администратор имеет доступ ко всем разделам сайта, имеет возможность контролировать, изменять и удалять учтенные данные пользователей (студентов и преподавателей) системы, удалять, изменять публикуемые на сайте новости и объявления, просматривать результаты тестирования и анкетирования, публиковать материалы в любых разделах сайта. Администратор имеет право и возможность заблокировать учетную запись любого пользователя системы в случае необходимости, а так же снять блокировку. Администратор определяет в системе какие дисциплины закреплены за каждым преподавателем с тем. чтобы последний мог редактировать только материалы, касающиеся своей дисциплины. Любой зарегистрированный пользователь, независимо от его права, должен иметь возможность изменения своих учетных данных (имени, пароля, и т.д., кроме ника).

Основные функции образовательного интернет сайта:

– тестирование;

– анкетирование;

– добавление информации (новости, объявления и др.);

– формирование статистики.

Начало работы программы:

1. Запускаем программу Ramus Educational. Далее создаем новый проект.

2. После создания проекта осуществляется запуск мастера проекта.

– на первом шаге в соответствующие поля вносим сведения об авторе, названии проекта и модели, а также выбираем тип нотации модели (IDEF0 или DFD);

– на втором шаге вводим название организации, использующей данный проект;

– на третьем шаге – даем краткое описание будущего проекта;

– четвертый шаг позволяет создать несколько основных классификаторов (в данном случае можно пропустить этот шаг). Так как модель программного обеспечения может хранить значительное количество объектов (статистика, преподаватели, расписание, материалы и т.д.), то в Ramus предусмотрена возможность упорядочено хранить информацию об этих объектах в виде системы классификаторов. Классификация объектов упрощает поиск и обработку информации об объектах модели, а так же и об объектах непосредственно не представленных на диаграммах процессов, но относящихся к процессам образовательного интернет сайта;

– на пятом шаге, предлагается выбрать классификаторы из созданных ранее (данный шаг пропускаем).

При необходимости можно завершить работу мастера.

После завершения работы мастера, отрывается рабочее пространство, в котором создается графическая модель. В верхней части приводятся сведения о проекте, введенные пользователем посредством мастера диаграмм.

Создание контекстной диаграммы:

1. На панели инструментов выбираем функциональный блок и устанавливаем его на рабочем пространстве.

2. Даем данному функциональному блоку имя «Сайт кафедры ИС».

3. Создаем стрелки на контекстной диаграмме согласно таблице 1.

модель управление сайт диаграмма

Таблица 1. Контекстная диаграмма

Наименование стрелки	**Тип**
Учебный план	Вход
Информация о преподавателе	Вход
Информация о студенте	Вход
Учебный материал	Вход
Информация о расписании	Вход
Бесплатные файлы ПО	Вход
Расписание	Выход
Статистика по студентам	Выход
Результаты тестирования	Выход
Результаты анкетирования	Выход
Объявления, новости	Выход
Материалы	Выход
Устав	Управление
Положение	Управление
Приказы	Управление
КТС	Механизм
Интернет	Механизм
Администратор	Механизм
Преподаватель	Механизм
Студент	Механизм

4. В результате получаем контекстную диаграмму (Приложение А)

Создание диаграммы декомпозиции:

1. В панели инструментов выбираем переход на нижний уровень, далее устанавливаем необходимое количество функциональных блоков (3), указываем тип диаграммы (IDEF0).

2. Автоматически будет создана диаграмма первого уровня декомпозиции с перенесенными в нее потоками родительской диаграммы. Осуществляем построение детализирующих диаграмм для функциональных блоков.

3. В результате получаем детализированную диаграмму первого уровня (Приложение А).

Создание диаграммы декомпозиции для блока А3

1. Нажимаем на блок А3, в панели инструментов выбираем переход на нижний уровень, далее устанавливаем необходимое количество функциональных блоков (5), указываем тип диаграммы (IDEF0).

5. Автоматически будет создана диаграмма второго уровня декомпозиции с перенесенными в нее потоками родительской диаграммы. Осуществляем построение детализирующих диаграмм для функциональных блоков.

6. В результате получаем детализированную диаграмму второго уровня (Приложение А).

Приложение А

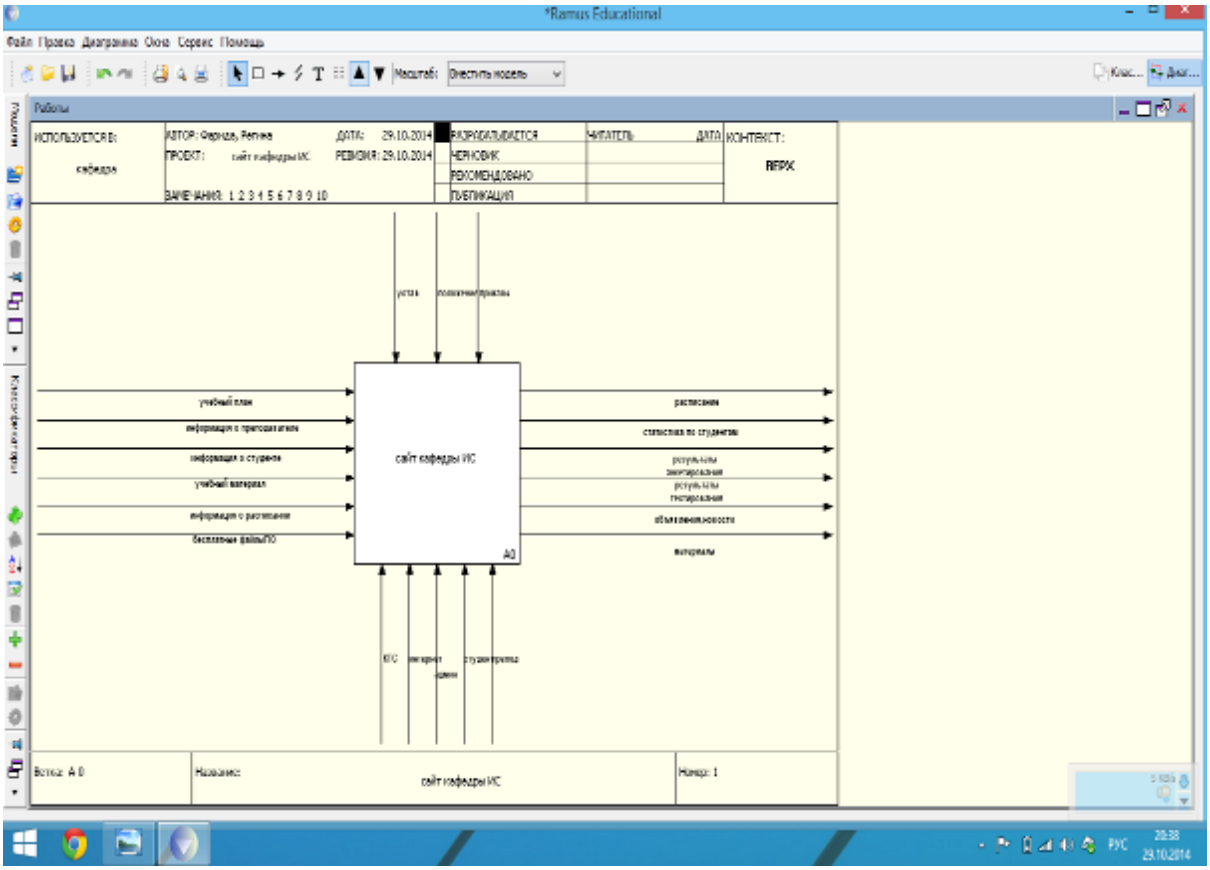


Рис. 1. Контекстная диаграмма «Сайт кафедры ИС».

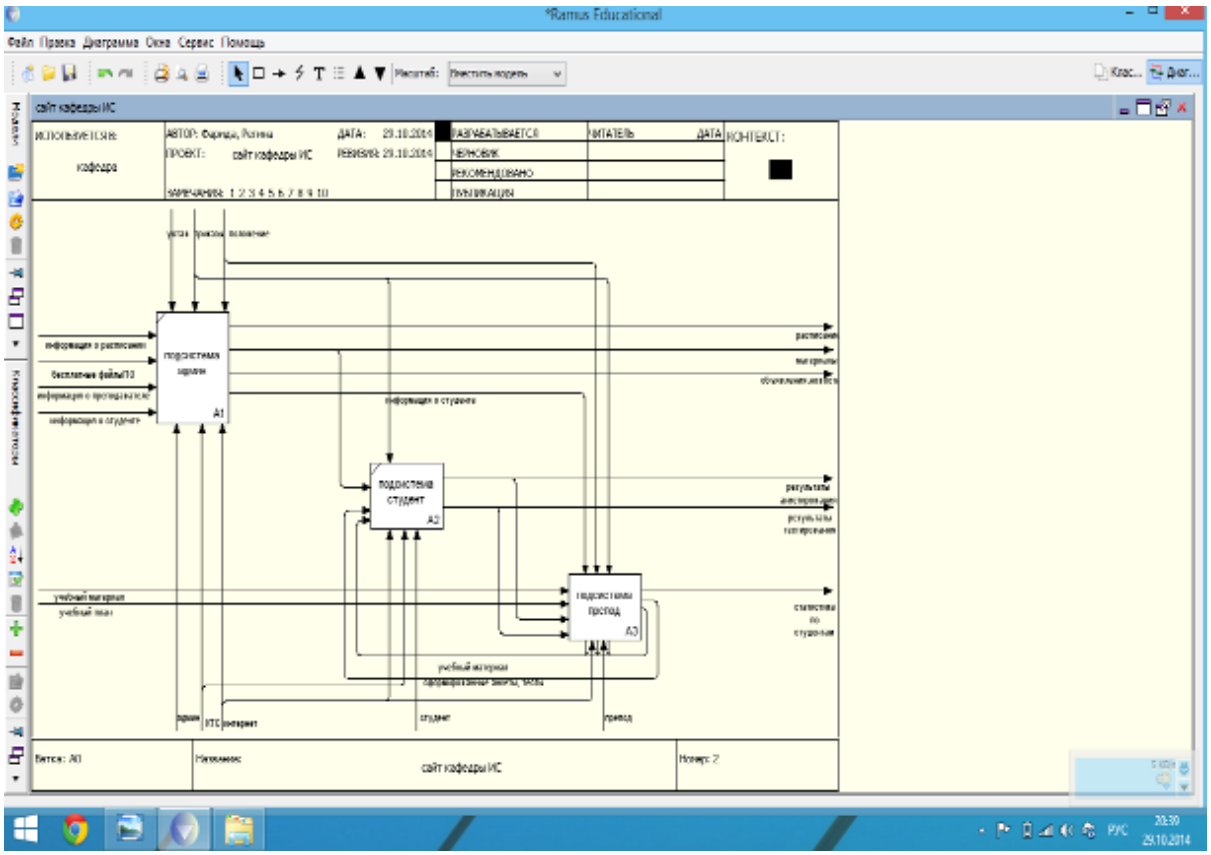


Рис. 2. Детализированная диаграмма первого уровня

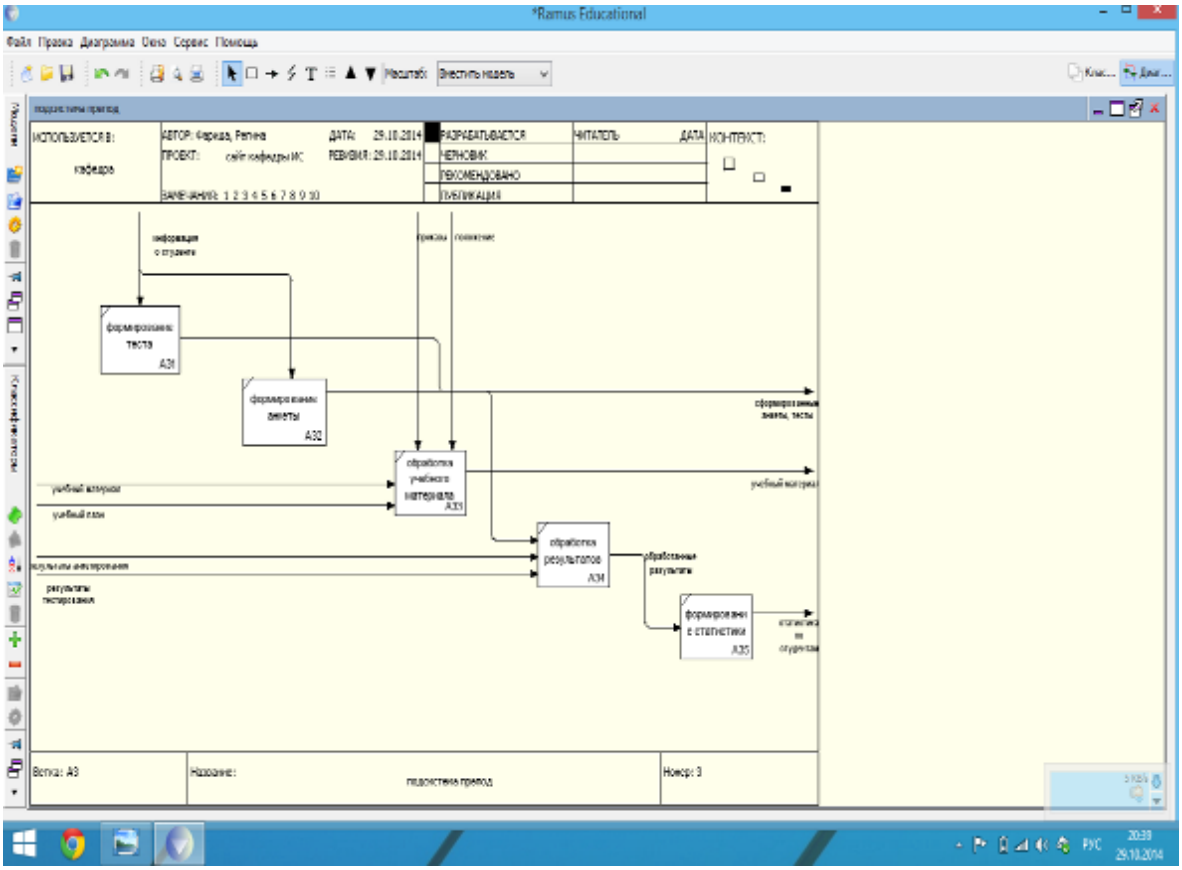


Рис. 3. Детализированная диаграмма второго уровня

4. Разработка DFD-модели в системе Ramus Educational.

Тема: Создание диаграммы потоков данных.

Цель: получить навык создания и редактирования функциональных моделей в программной среде Ramus.

Программное обеспечение: RAMUS Educational.

Порядок выполнения

1. Создайте новый проект. При редактировании свойств проекта укажите нотацию DFD.

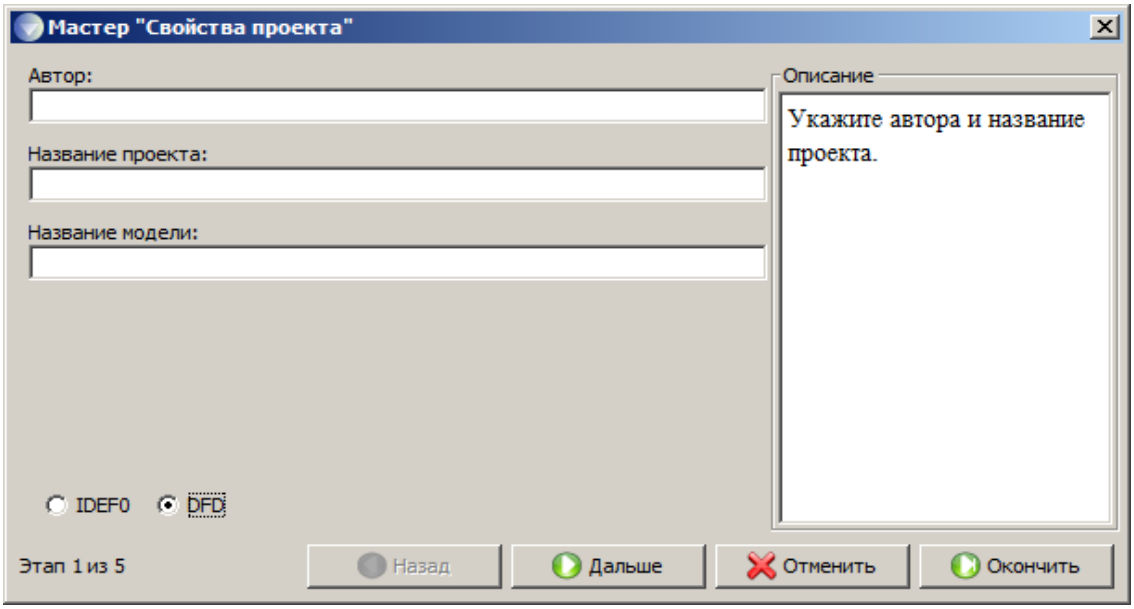


Рис. 1 - Редактирование свойств проекта.

2. Введите названия классификаторов (элементы DFD: внешняя сущность и накопитель данных) в соответствии с рисунком 2.

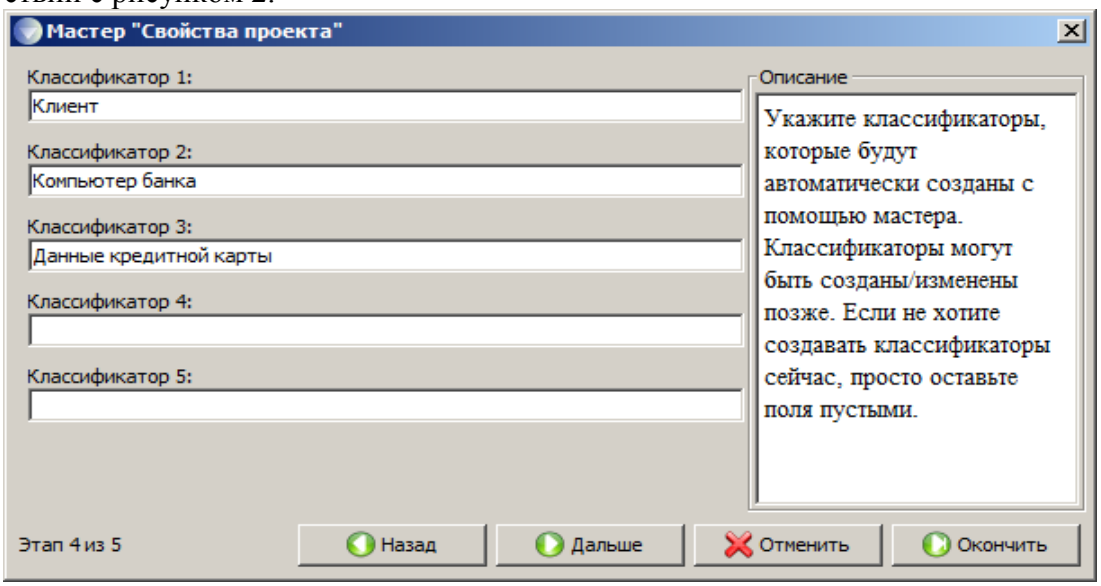


Рис. 2 - Создание классификаторов.

3. С помощью панели элементов для диаграммы DFD создайте диаграмму, как показано на рисунке 2.

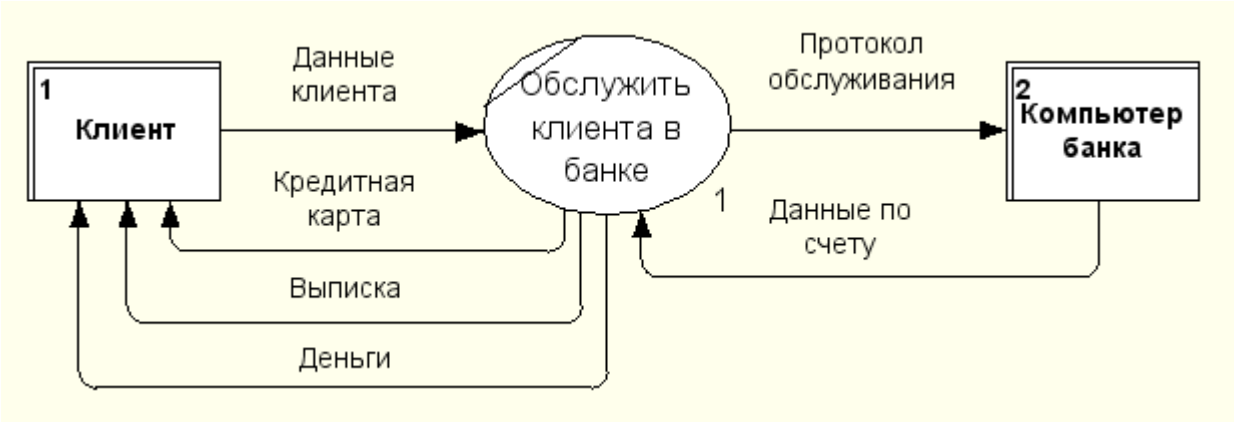


Рис. 3 - Контекстная диаграмма DFD

4. Выполните декомпозицию диаграммы DFD на 4 процесса.
5. Создайте диаграмму по рисунку 3.
6. По ниже приведенным вариантам создайте диаграмму DFD.

Варианты заданий DFD

Вариант 1.

Создать диаграмму потоков данных процесса «ОБСЛУЖИТЬ» при работе библиотекаря с клиентами, начиная работу с заказа клиентом нужного ему издания из хранилища.

Вариант 2.

Создать диаграмму потоков данных процесса «ПРОВЕСТИ МАРКЕТИНГОВЫЕ ИССЛЕДОВАНИЯ», подробно рассмотрев все процессы, происходящие при этом. В качестве внешних сущностей можно выбрать «КЛИЕНТ» и «РЫНОК».

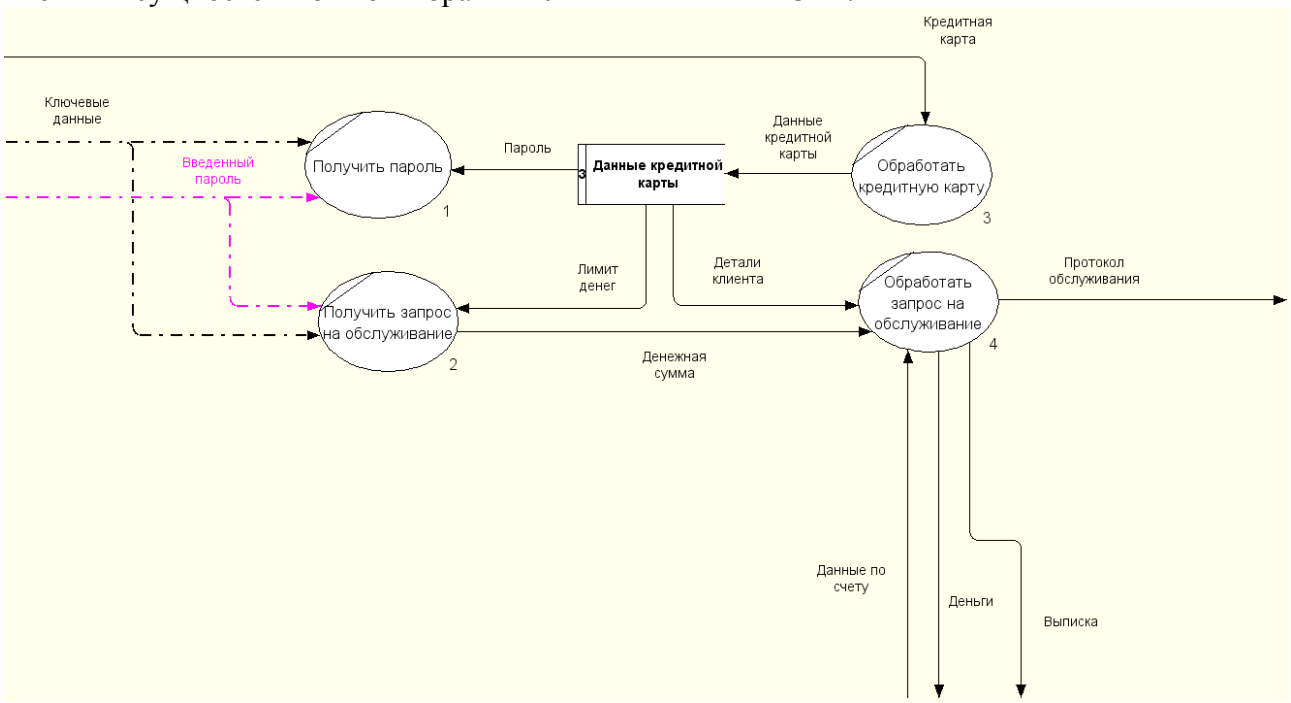


Рис. 4 - Диаграмма декомпозиции DFD

Вариант 3.

Создать диаграмму потоков данных процесса «ОБЕСПЕЧИТЬ ПРЕДПРЯТИЕ ТОВАРОМ» при работе отдела сбыта с поставщиками.

Вариант 4.

Создать диаграмму потоков данных процесса "ПЛАНИРОВАТЬ ДЕЯТЕЛЬНОСТЬ ПРЕДПРИЯТИЯ», учитывая финансовую, хозяйственную и прочие деятельности предприятия.

Вариант 5.

Создать диаграмму потоков данных процесса «СОЗДАТЬ ПРОГРАММУ» при работе программиста над разработкой и созданием ПО.

Вариант 6.

Создать диаграмму потоков данных процесса «РАЗРАБОТАТЬ КОНСАЛТИНГОВЫЙ ПРОЕКТ», учитывая основные этапы при проведении консалтинга:

1. анализ первичных требований;
2. проведение обследования деятельности предприятия;
3. построение моделей «как есть» и «как должно быть»;
4. оценка эффективности деятельности предприятия;
5. реорганизация деятельности;
6. разработка системного проекта;
7. разработка предложений по автоматизации;
8. выбор, разработка и внедрение новой информационной системы. Создать словарь данных, описав все хранилища данных и внешние сущности.

Вариант 7.

Создать диаграмму потоков данных процесса «ОБЕСПЕЧИТЬ ПРОДАЖУ ТОВАРА» при работе отдела сбыта крупного предприятия, работающего как на местном рынке, так и на мировом.

Вариант 8.

Создать диаграмму потоков данных процесса «РАБОТА ТУРИСТИЧЕСКОЙ ФИРМЫ» при работе проектировщика с использованием компьютера.

Вариант 9.

Создать диаграмму потоков данных процесса «ОБСЛУЖИТЬ» при покупке акций на рынке ценных бумаг.

5. Два подхода к построению DFD-моделей.

Диаграммы DFD можно строить с использованием подхода, аналогичного структурному методу анализа и проектирования, применяемому в IDEFO. Вначале строится модель физической реализации реальной системы, которая используется пользователями в настоящее время. Затем создается логическая модель текущего состояния системы для моделирования основных требований существующей системы. После этого создается новая логическая модель для отражения основных параметров предлагаемой разрабатываемой системы. Наконец, создается новая физическая модель, реализующая логическую модель новой системы.

В настоящее время при разработке информационных систем завоевывает все большую популярность альтернативный подход, известный как разделение событий, в котором для моделирования системы строится несколько моделей DFD. Вначале строится логическая модель, отображающая систему как набор действий и описывающая, что должна делать система. Затем строится модель окруэюения, описывающая систему как объект, отвечающий на события, порождаемые внешними сущностями. Такая модель обычно состоит из описания назначения системы, одной диаграммы контекстного уровня и списка событий. Контекстная диаграмма содержит один функциональный блок, представляющий систему в целом, и внешних сущностей (окружения), с которыми система взаимодействует.

На заключительном этапе создается модель поведения, показывающая, как система обрабатывает те или иные события. Эта модель начинается с единственной диаграммы с одним функциональным блоком на каждый ответ системы на событие, описанное в модели окружения. Хранилища данных в модели поведения используются для моделирования данных, которые должны сохраняться в промежутках между обработкой событий. Потоки применяются для соединения элементов диаграмм между собой и для проверки согласованности моделей поведения и окружения.

При подготовке такого рода моделей к различным презентациям обычно необходима их "чистка". При этом может применяться как создание упрощенных родительских диаграмм посредством объединения нескольких функциональных блоков в один, так и декомпозиция некоторых элементов для более ясного восприятия модели.

В DFD каждый номер функционального блока может включать в себя префикс, номер родительской диаграммы и собственно номер объекта (рис. 1). Номер объекта уникальным образом идентифицирует функциональный блок на диаграмме. Номер родительской диаграммы и номер объекта в совокупности обеспечивают уникальную идентификацию каждого блока модели.

Уникальные номера присваиваются также каждому хранилищу данных и каждой внешней сущности вне зависимости от расположения. Каждый номер хранилища содержит префикс D (от английского Data Store) и уникальный номер хранилища в модели (например, D3).

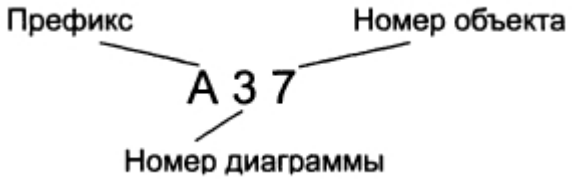


Рис. 1. Компоненты номер функционального блока DFD

Аналогично каждый номер каждой внешней сущности содержит префикс E (от английского External entity) и уникальный номер сущности в модели (например, E5).

Смоделировать работу системы технического обслуживания воздушного судна по указанию преподавателя.

Что такое DFD?

Какие новые типы объектов может содержать DFD?

Какие физические характеристики системы отражаются в DFD - моделях? Область применения DFD.

Привести пример контекстной DFD - диаграммы.

Назначение функциональных блоков, внешних сущностей и стрелок (потоков данных) в DFD.

Какова функция хранилищ данных в производственных и информационных системах?

Привести пример разветвления и объединения стрелок.

Каковы основные подходы к построению DFD - моделей?

Принципы нумерации объектов в DFD.

6. Изучить предметную область. Спроектировать с учетом особенностей данной предметной области диаграмму бизнес-процессов (activity diagrams), используя Case-средство Ramus.

Цель работы:

Изучить предметную область. Спроектировать с учетом особенностей данной предметной области диаграмму бизнес-процессов (activity diagrams), используя Case-средство Ramus.

Рабочее задание:

Используя средство проектирования Ramus, отобразить на диаграммах процессы предметной области и произвести декомпозицию.

Описание предметной области:

Сервисный центр «Мега» — организация, занимающаяся оказанием услуг по сервисной поддержке и обслуживанию цифровой и компьютерной техники. Основное направление его деятельности включает предторговый, гарантийный и послепродажный ремонт, а также обслуживание по установке технического и программного обеспечения.

Оформлением в ремонт и выдачу уже отремонтированной техники в сервисном центре занимается секретарь центра. Ремонт техники осуществляет мастер по ремонту.

При оформлении техники в ремонт, секретарь заполняет заявку на ремонт. В ней указывается: информация о клиенте (фамилия, имя, отчество, адрес регистрации, контактный телефон, паспортные данные, вид и наименование техники, перечень неисправностей).

Секретарь регистрирует заявку в журнале заявок, присваивая ей номер и указывая дату приема техники в ремонт. Образовавшийся номер присваивает технике и наклеивается на корпус техники. Секретарь, вместе с клиентом, осматривает технику на наличие физических повреждений. Это необходимо для того, чтобы не возникли проблемы при возврате техники.

Копию заявки также приклеивают к корпусу техники, и передают вместе с документами (пакет документов, оформленный при продаже) и гарантийным талоном в мастерскую. Оригинал заявки заказчик оставляет у себя.

Техника передается мастеру центра «Мега», который, находясь на своём рабочем месте, знакомится с предоставленным пакетом документов, гарантийным талоном, заявлением. Просматривает «список неисправностей», указанных заказчиком.

Мастер проводит диагностику техники для определения неисправностей. Также мастер должен определить, является ли случай гарантийным.

Если в результате диагностики, выявлена неисправность и сделано заключение, что данная поломка произошла не по вине заказчика, а из-за производственного брака (или по другой причине), то мастер извлекает не исправную деталь и заполняет требование на закупку (поставку) детали. В требовании указывается серийный номер детали, полное наименование, номер гарантии.

Требование просматривается, подписывается директором сервисного центра и, далее, секретарь регистрирует требование в журнале требований на комплектующие и отправляет представителю производителя данной техники, который и предоставляет на нее гарантию.

7. Моделирование управленческого учета на предприятии.

В общем виде система управленческого учета выглядит следующим образом:

Блок бухгалтерского (финансового) учета Блок управленческого производственного учета Блок учета управленческих решений Блок стратегического управленческого учета.

В состав блока управленческого производственного учета входят следующие модули, реализующие создание информационной базы для принятия комплексных управленческих решений в рамках оптимального функционирования коммерческой фирмы:

1. модуль эффективности коммерческой сделки и ценообразования;
2. модуль планирования;
3. модуль запасов;
4. модуль анализа финансового состояния (АФС);
5. модуль документооборота;
6. модуль капиталодвижения;

7. модуль платежного графика;
8. модуль расчета рисков коммерческих проектов;
9. модуль функционально-стоимостного анализа (ФСА);
10. модуль бухгалтерского (финансового) учета.

В модуле эффективности коммерческой сделки и ценообразования имеются три подмодуля. Первый — подмодуль формирования кода проекта и калькулирования постоянных (накладных) затрат. В первом подмодуле после импорта данных бухгалтерского учета (наиболее эффективным является подход использования данных и плана счетов бухгалтерского учета, реализующий все возможные варианты движения денежных средств) осуществляется на базе аналитических данных бухгалтерского учета формирование кода проекта, содержащего наименования ассортиментной единицы, кода контрагента, даты совершения операции. Таким образом, первоначально мы получаем данные о частях структуры цены: переменных затрат и норме прибыли. Постоянные затраты структуры цены продукта мы получим на основе подсчета удельных весов ассортиментной единицы (сумма ее переменных издержек в процентах от совокупной суммы переменных издержек всего ассортиментного плана коммерческой фирмы (можно определить и на основе удельного веса выручки ассортиментной единицы от общего объема выручки коммерческой фирмы). Таким образом, получается полная калькуляция цены каждой ассортиментной единицы и уже во втором подмодуле производится расчет рентабельности каждой ассортиментной единицы. В третьем подмодуле первого модуля производится CVP анализ каждой ассортиментной единицы, вида бизнеса, определение бизнес направлений фирмы в целом (CVP анализ и анализ ассортиментной политики в рамках всего ассортиментного плана).

В модуле планирования существуют так же три подмодуля. Первый подмодуль — модуль расчета хронометражных нормативов. Здесь фиксируются хронометражные данные в рамках плановых нормативных выполнений каждой бизнес операции в составе единого комплексного бизнес процесса фирмы. Эти данные необходимы для построения нормативного сетевого графика, который строится в третьем подмодуле, называемом «единая ось бизнес- событий», где объединяются последовательно все бизнес- операции коммерческой компании. Второй подмодуль посвящен расчету и учету образования очередей вследствие несоответствия функций спроса и предложения. Реализуются расчеты на основе математической модели СМО с ожиданием и ограниченной длиной очереди. Производится расчет мощностного баланса фирмы.

Модуль запасов включает три подмодуля:

1. подмодуль товарных запасов;
2. подмодуль запасов готовой продукции;
3. подмодуль денежного запаса.

Все три модуля строятся на основе математической модели движения запасов с построением графиков и расчета трех видов издержек:

• подготовительно-заключительного времени;
• издержек хранения;
• издержек дефицита.

Модуль анализа финансового состояния включает построение графиков безубыточности по каждой ассортиментной единице, виду бизнеса и компании в целом.

В рамках модуля документооборота производится формирование всей базы документооборота, существующего в фирме по всем бизнес функциям, отделам и подразделениям с учетом как документов финансового, так и не финансового характера (включая договора, внутрицеховые документы, распоряжения и т.д.). На основе документооборота и планового сетевого графика происходит формирования функциональных должностных инструкций.

Модуль капиталодвижения содержит три подмодуля:

1. подмодуль кредитных сделок, включающий ТЭО кредитного проекта:

2. подмодуль дивидендной политики;

3. подмодуль акционерного капитала.

Все три подмодуля содержат расчеты движения капитала, его эффективности с учетом интересов сторон, осуществляющих это движение.

Модуль платежного графика является сигнальной системой, служащей для оперативного анализа движения денежных средств с точки зрения временных характеристик, как поступления, так и расходования денежных средств, важнейшей частью которого является график налоговых выплат.

В модуле рисков производится подсчет на основе математического аппарата рисков важнейших бизнес -событий фирмы.

Модуль функционально-стоимостного анализа характеризуется необходимостью получения информации по тем операциям, которые первично, формально не носят стоимостного характера, однако, имеют важнейшее значение с точки зрения принятия перспективных управленческих решений (например: вероятная дата поставки оборудования с точки зрения подготовки цеховых площадей, персонала и дополнительных условий).

Важнейшим является не только осуществление функций каждого модуля, но и существование функциональной межмодульной связи (так, например: модуль платежного графика должен иметь связь с подмодулем денежных запасов модуля запасов).

Обмен информацией блоков бухгалтерского (финансового) учета и управленческого производственного учета является важнейшим, с точки зрения поставки информации в рамках проведенных операций, и тех требований, которые предъявляет управленческий учет к функционированию системы бухгалтерского (финансового) учета. На основе своевременного калькулирования постоянных затрат (с точки зрения момента реализации готовой продукции), и на основе данных бухгалтерской аналитики производится формирование кода проекта.

В рамках третьего блока системы управленческого учета – блока учета управленческих решений предусматривает разработку системы формирования решений-. В рамках модуля реализуется создание комплекса (дерева) управленческих решений, характеризующим всю систему бизнес-событий фирмы (последовательность, взаимосвязь, временные характеристики с учетом постановки, контроля исполнения, анализа и архивирования и т.д.) Разработанные управленческие решения функционирования бизнеса фирмы имеют модельную расчетную реализацию в рамках обмена с блоком управленческого производственного учета.

Четвертый блок стратегического управленческого учета посвящен перспективному позиционированию коммерческой фирмы (:разработка стратегических планов фирмы, учет этапа жизненного цикла в политике ценообразования ассортиментной единицы, решение проблем ресурсного обеспечения в долговременной перспективе и развития бизнеса в региональном и контрагенском аспекте и т.д.

Главная проблема в создании подобной модели — тщательная разработка проблемно ориентированного алгоритма каждого модуля с учетом межблочного информационного взаимодействия на основе всех изложенных проблемных моментов.

8. Ознакомление с основными элементами определения, представления, проектирования и моделирования информационных систем с помощью языка UML.

Существует множество технологий и инструментальных средств, с помощью которых можно реализовать в некотором смысле оптимальный проект ИС, начиная с этапа анализа и заканчивая созданием программного кода системы. В большинстве случаев эти технологии предъявляют весьма жесткие требования к процессу разработки и используемым ресурсам, а попытки трансформировать их под конкретные проекты оказываются безуспешными. Эти технологии представлены CASE-средствами верхнего уровня или

CASE-средствами полного жизненного цикла (upper CASE tools или full life-cycle CASE tools). Они не позволяют оптимизировать деятельность на уровне отдельных элементов проекта, и, как следствие, многие разработчики перешли на так называемые CASE-средства нижнего уровня (lower CASE tools). Однако они столкнулись с новой проблемой — проблемой организации взаимодействия между различными командами, реализующими проект.

Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML) явился средством достижения компромисса между этими подходами. Существует достаточное количество инструментальных средств, поддерживающих с помощью *UML*жизненный цикл информационных систем, и, одновременно, *UML* является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

Создание UML началось в октябре 1994 г., когда Джим Рамбо и Гради Буч из Rational Software Corporation стали работать над объединением своих методов OMT и Booch. В настоящее время консорциум пользователей UML Partners включает в себя представителей таких грандов информационных технологий, как Rational Software, Microsoft, IBM, Hewlett-Packard, Oracle, DEC, Unisys, IntelliCorp, Platinum Technology.

UML представляет собой *объектно-ориентированный* язык моделирования, обладающий следующими основными характеристиками:

- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;
- содержит механизмы расширения и специализации базовых концепций языка.

UML — это стандартная нотация визуального моделирования программных систем, принятая консорциумом Object Managing Group (OMG) осенью 1997 г., и на сегодняшний день она поддерживается многими объектно-ориентированными CASE-продуктами.

UML включает внутренний набор средств моделирования, которые сейчас приняты во многих методах и средствах моделирования. Эти концепции необходимы в большинстве прикладных задач, хотя не каждая концепция необходима в каждой части каждого приложения. Пользователям языка предоставлены возможности:

- строить модели на основе средств ядра, без использования механизмов расширения для большинства типовых приложений;
- добавлять при необходимости новые элементы и условные обозначения, если они не входят в ядро, или специализировать компоненты, систему условных обозначений (нотацию) и ограничения для конкретных предметных областей.

Стандарт UML предлагает следующий набор диаграмм для моделирования:

- диаграммы вариантов использования (use case diagrams) – для моделирования бизнес-процессов организации и требований к создаваемой системе);
- диаграммы классов (class diagrams) – для моделирования статической структуры классов системы и связей между ними;
- диаграммы поведения системы (behavior diagrams):
- диаграммы взаимодействия (interaction diagrams):
- диаграммы последовательности (sequence diagrams) и
- кооперативные диаграммы (collaboration diagrams) – для моделирования процесса обмена сообщениями между объектами;

- диаграммы состояний (statechart diagrams) – для моделирования поведения объектов системы при переходе из одного состояния в другое;

- диаграммы деятельностей (activity diagrams) – для моделирования поведения системы в рамках различных вариантов использования, или моделирования деятельностей;

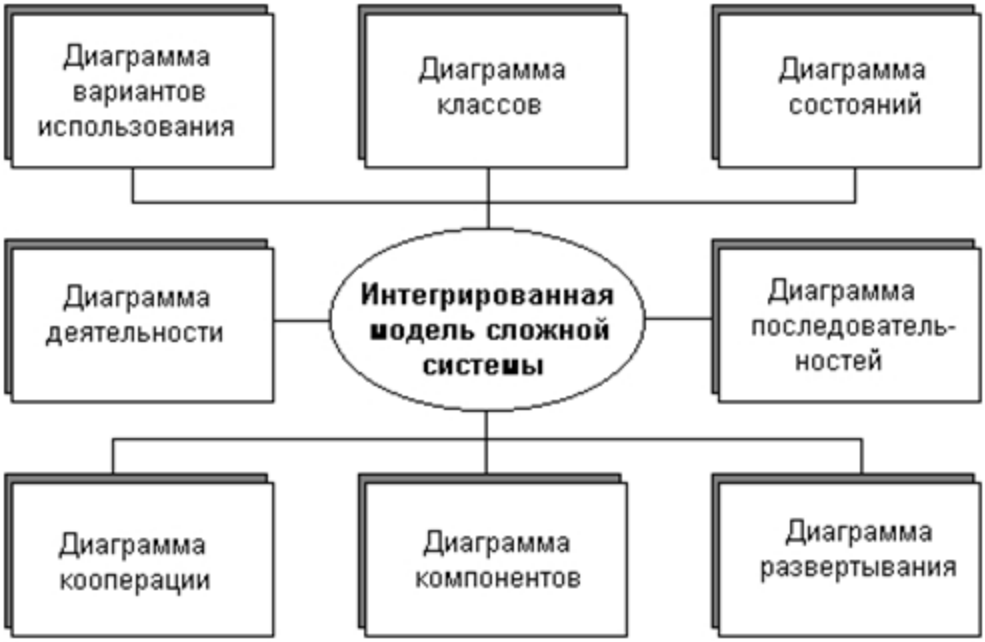


Рис. 1. Интегрированная модель сложной системы в нотации языка UML.

- диаграммы реализации (implementation diagrams):

- диаграммы компонентов (component diagrams) – для моделирования иерархии компонентов (подсистем) системы;

- диаграммы развертывания (deployment diagrams) – для моделирования физической архитектуры системы.

Диаграммы вариантов использования

Понятие варианта использования (use case) впервые ввел Ивар Якобсон и придал ему такую значимость, что в настоящее время вариант использования превратился в основной элемент разработки и планирования проекта.

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать. На языке UML вариант использования изображают следующим образом:

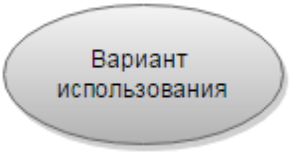


Рис.2. Вариант использования.

Действующее лицо (actor) – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Несмотря на то, что на диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок, действующее лицо может также быть внешней системой, которой необходима некоторая информация от данной системы. Показывать на диаграмме действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования. На языке UML действующие лица представляют в виде фигур:

Рис.3. Действующее лицо

Действующие лица делятся на три основных типа:
- пользователи;
- системы;
- другие системы, взаимодействующие с данной;
- время.

Время становится действующим лицом, если от него зависит запуск каких-либо событий в системе.

Связи между вариантами использования и действующими лицами

В языке UML на диаграммах вариантов использования поддерживается несколько типов связей между элементами диаграммы. Это связи коммуникации (communication), включения (include), расширения (extend) и обобщения (generalization).

Связь коммуникации – это связь между вариантом использования и действующим лицом. На языке UML связи коммуникации показывают с помощью однонаправленной ассоциации (сплошной линии).

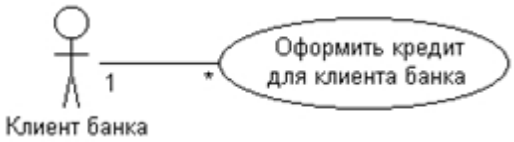


Рис.4. Пример связи коммуникации

Связь включения применяется в тех ситуациях, когда имеется какой-либо фрагмент поведения системы, который повторяется более чем в одном варианте использования. С помощью таких связей обычно моделируют многократно используемую функциональность.

Связь расширения применяется при описании изменений в нормальном поведении системы. Она позволяет варианту использования только при необходимости использовать функциональные возможности другого.

С помощью связи обобщения показывают, что у нескольких действующих лиц имеются общие черты.

Диаграммы состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий.

Существует много форм диаграмм состояний, незначительно отличающихся друг от друга семантикой.

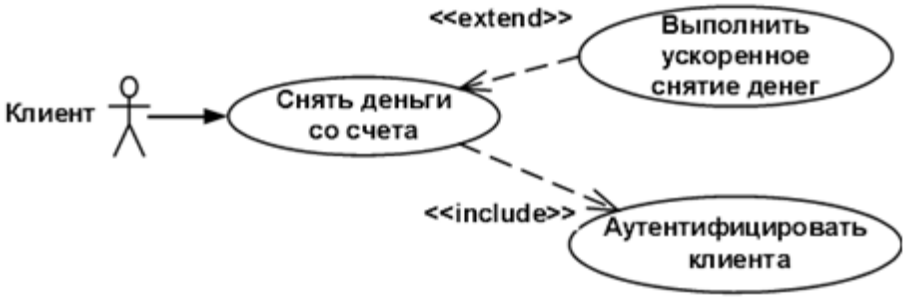


Рис.5. Пример связи включения и расширения.

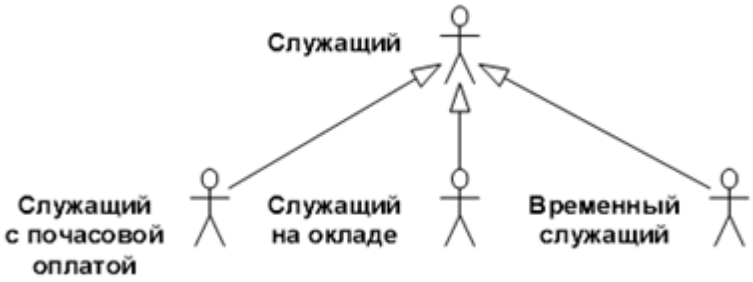


Рис.6. Пример связи обобщения

На диаграмме имеются два специальных состояния – начальное (start) и конечное (stop). Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан. Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. На диаграмме состояний может быть одно и только одно начальное состояние. В то же время, может быть столько конечных состояний, сколько вам нужно, или их может не быть вообще. Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Процессы, происходящие, когда объект находится в определенном состоянии, называются действиями (actions).

С состоянием можно связывать данные пяти типов: деятельность, входное действие, выходное действие, событие и история состояния.

Деятельность

Деятельностью (activity) называется поведение, реализуемое объектом, пока он находится в данном состоянии. Деятельность – это прерываемое поведение. Оно может выполняться до своего завершения, пока объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Деятельность изображают внутри самого состояния, ей должно предшествовать слово do (делать) и двоеточие.

Входное действие

Входным действием (entry action) называется поведение, которое выполняется, когда объект переходит в данное состояние. Данное действие осуществляется не после того, как объект перешел в это состояние, а, скорее, как часть этого перехода. В отличие от деятельности, входное действие рассматривается как непрерываемое. Входное действие также показывают внутри состояния, ему предшествует слово entry (вход) и двоеточие.

Выходное действие

Выходное действие (exit action) подобно входному. Однако, оно осуществляется как составная часть процесса выхода из данного состояния. Оно является частью процесса такого перехода. Как и входное, выходное действие является непрерываемым.

Выходное действие изображают внутри состояния, ему предшествует слово exit (выход) и двоеточие.

Поведение объекта во время деятельности, при входных и выходных действиях может включать отправку события другому объекту. В этом случае описанию деятельности, входного действия или выходного действия предшествует знак « ^ ».

Соответствующая строка на диаграмме выглядит как

Do: ^Цель.Событие (Аргументы)

Здесь Цель – это объект, получающий событие, Событие – это посылаемое сообщение, а Аргументы являются параметрами посылаемого сообщения.

Деятельность может также выполняться в результате получения объектом некоторого события. При получении некоторого события выполняется определенная деятельность.

Переходом (Transition) называется перемещение из одного состояния в другое. Совокупность переходов диаграммы показывает, как объект может перемещаться между своими состояниями. На диаграмме все переходы изображают в виде стрелки, начинающейся на первоначальном состоянии и заканчивающейся последующим.

Переходы могут быть рефлексивными. Объект может перейти в то же состояние, в котором он в настоящий момент находится. Рефлексивные переходы изображают в виде стрелки, начинающейся и завершающейся на одном и том же состоянии.

У перехода существует несколько спецификаций. Они включают события, аргументы, ограждающие условия, действия и посылаемые события.

События

Событие (event) – это то, что вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода.

На диаграмме для отображения события можно использовать как имя операции, так и обычную фразу.

Большинство переходов должны иметь события, так как именно они, прежде всего, заставляют переход осуществиться. Тем не менее, бывают и автоматические переходы, не имеющие событий. При этом объект сам перемещается из одного состояния в другое со скоростью, позволяющей осуществиться входным действиям, деятельности и выходным действиям.

Ограждающие условия

Ограждающие условия (guard conditions) определяют, когда переход может, а когда не может осуществиться. В противном случае переход не осуществится.

Ограждающие условия изображают на диаграмме вдоль линии перехода после имени события, заключая их в квадратные скобки.

Ограждающие условия задавать необязательно. Однако если существует несколько автоматических переходов из состояния, необходимо определить для них взаимно исключающие ограждающие условия. Это поможет читателю диаграммы понять, какой путь перехода будет автоматически выбран.

Действие

Действием (action), как уже говорилось, является непрерываемое поведение, осуществляющееся как часть перехода. Входные и выходные действия показывают внутри состояний, поскольку они определяют, что происходит, когда объект входит или выходит из него. Большую часть действий, однако, изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния.

Действие рисуют вдоль линии перехода после имени события, ему предшествует косая черта.

Событие или действие могут быть поведением внутри объекта, а могут представлять собой сообщение, посылаемое другому объекту. Если событие или действие посылается другому объекту, перед ним на диаграмме помещают знак « ^ ».

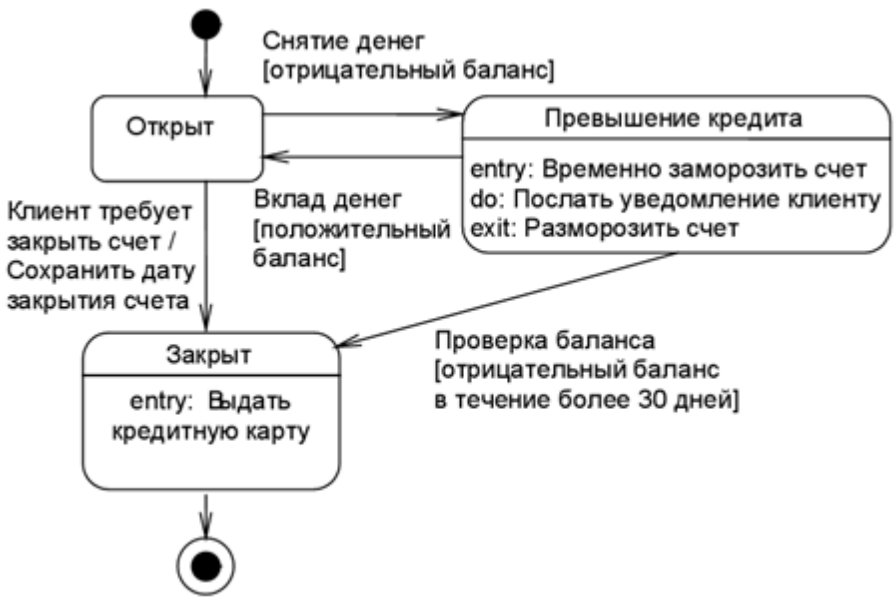


Рис. 7. Пример диаграммы состояний.

4. Порядок выполнения работы

1. Изучить предлагаемый теоретический материал.

2. Постройте диаграмму вариантов использования для выбранной информационной системы.

4. Постройте диаграмму состояний для конкретных объектов информационной системы.

5. Оформите отчёт.

5. Содержание отчета

В отчете следует указать:

1. Цель работы
2. Введение
3. Программные средства, используемые при выполнении работы.
4. Основную часть (описание самой работы), выполненную согласно требованиям.
5. Заключение (выводы).
6. Список используемых источников.

9. Создание диаграммы вариантов использования с помощью ArgoUML.

Цель работы

Научиться определять требования к программной системе с помощью построения диаграммы вариантов использования языка UML.

Вариант использования (use case) представляет собой общую спецификацию совокупности выполняемых системой действий с целью предоставления некоторого наблюдаемого результата, который имеет значение для одного или нескольких актеров.

Другими словами, вариант использования представляет собой законченный фрагмент поведения системы с точки зрения тех или иных заинтересованных лиц. Хотя каждый вариант использования предполагает реализацию системой некоторой последовательности действий, которые должны быть выполнены с целью предоставления соответствующего результата или сервиса, сами эти действия на диаграмме вариантов использования никак не изображаются.

Общепринятым графическим изображением варианта использования в нотации языка UML является эллипс, внутри которого или ниже его записывается имя в форме строки текста. Пример варианта использования представлен на рис. 1.

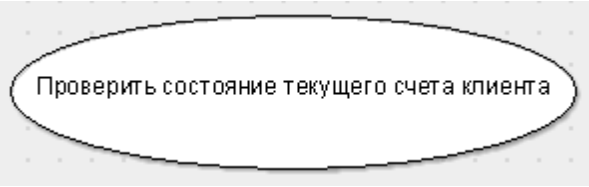


Рис. 1. Стандартное графическое обозначение варианта использования.

Примерами вариантов использования могут являться следующие фрагменты законченного функционального поведения системы: получение справки о состоянии счета клиента, оформление заказа на покупку товара, идентификация пользователя при входе в систему, отображение графической формы на экране монитора и другие поведения рассматриваемых систем.

2. ***Актер (actor)*** представляет собой внешнюю по отношению к проектируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач.

Формально в контексте языка UML 2.0 каждый актер специфицирует некоторую роль, которую играет пользователь или любая другая система, взаимодействующая с субъектом. Термин «роль» здесь используется неформально.

Примерами актеров могут быть: кассир, клиент банка, банковский служащий, президент, продавец магазина, менеджер отдела продаж, пассажир авиарейса, водитель автомобиля, администратор гостиницы, сотовый телефон, налоговая инспекция и другие сущности, имеющие отношение к концептуальной модели соответствующей предметной области.

3. ***Субъектом (subject)*** в контексте языка UML называется любой элемент модели, который обладает функциональным поведением.

Субъект вариантов использования может быть некоторой физической системой или любым другим элементом с поведением, таким как подсистема, класс или компонент. Графически субъект на диаграмме вариантов использования изображается в форме прямоугольника, внутри которого размещаются отдельные варианты использования. Каждый субъект имеет имя, которое должно соответствовать имени элемента моделируемой системы, владеющего данным поведением.

4. ***Комментарий (comment)*** в языке UML 2.0 предназначен для включения в модель произвольной текстовой в форме примечания, которое может быть присоединено к одному или нескольким элементам разрабатываемой модели. В качестве такой информации могут быть, например, пояснения разработчика относительно назначения элементов диаграммы, рекомендации руководителя проекта по реализации модели, справочная информация об авторе и особенностях разработки отдельных элементов модели. Применительно к диаграммам вариантов использования комментарий может иметь уточняющую информацию, относящуюся к контексту тех или иных вариантов использования или актеров.

Графически комментарий на всех типах диаграмм изображается в форме прямоугольника с «загнутым» правым верхним уголком (рис. 2).

Отношения на диаграмме вариантов использования

Под ***отношением (relationship)*** в языке UML понимается произвольная семантическая взаимосвязь между отдельными элементами модели.

Виды отношений на диаграммах вариантов использования:

1. ***Ассоциация (association)*** является одним из фундаментальных понятий в языке UML и может использоваться на различных канонических диаграммах при построении визуальных моделей. Применительно к диаграммам вариантов использования отношение ассоциации может служить только для обозначения взаимодействия актера с вариантом использования.

На диаграммах вариантов использования, так же как и на других диаграммах, отношение ассоциации обозначается сплошной линией, соединяющей актера и вариант использования. При этом следует различать *направленные* (рис. 3) и *ненаправленные* (рис. 4) *ассоциации*.

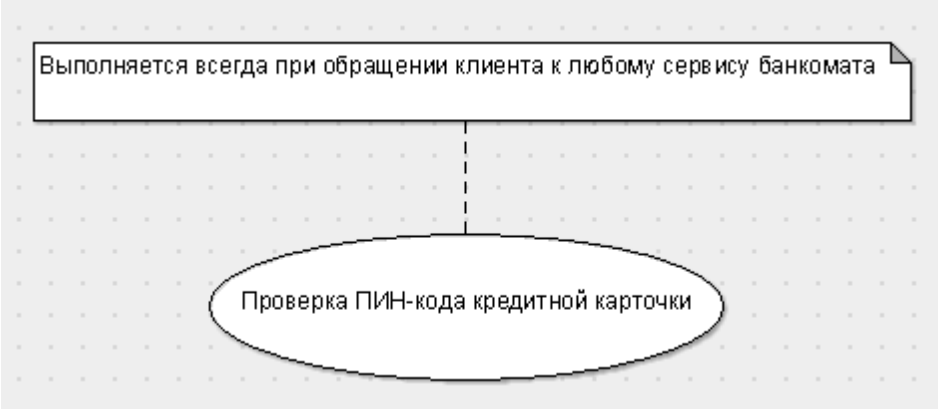


Рис. 2. Пример комментария в языке UML 2.0

Рис. 3. Пример графического представления отношения направленной ассоциации между актером и вариантом использования.

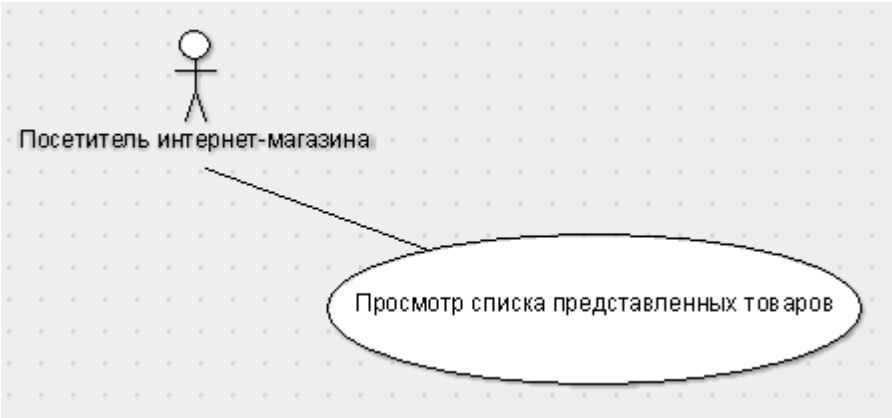


Рис. 5. Пример графического представления отношения ненаправленной ассоциации между актером и вариантом использования.

В контексте диаграммы вариантов использования направленная ассоциация от актера к варианту использования может указывать на тот факт, что данный актер является инициатором выполнения соответствующего варианта использования. Направленная ассоциация от варианта использования к актеру может указывать также на тот факт, что

данному актеру предоставляется справочная или отчетная информация о результатах функционирования моделируемой системы.

2. *Отношение зависимости (dependency)* определяется как форма взаимосвязи между двумя элементами модели, предназначенная для спецификации того обстоятельства, что изменение одного элемента модели приводит к изменению некоторого другого элемента.

Отношение включения (include) является частным случаем отношения зависимости и специфицирует тот факт, что некоторый вариант использования содержит поведение, определенное в другом варианте использования.

Графически данное отношение обозначается как отношение зависимости в форме пунктирной линии с «V»-образной стрелкой, направленной от зависимого варианта использования к независимому варианту использования (рис. 5). Зависимый вариант использования часто называют также *базовым* или *включающим*, а независимый – *включаемым* вариантом использования. При этом линия со стрелкой обязательно помечается ключевым словом <<include>>, записанным в форме стереотипа.

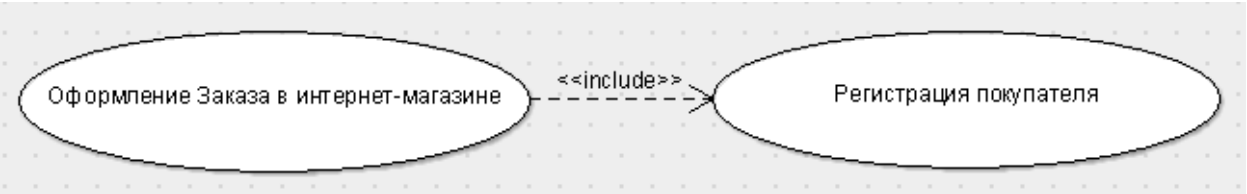


Рис. 5. Пример графического изображения отношения включения между вариантами использования

Отношение расширения (extend) является частным случаем отношения зависимости и определяет взаимосвязь одного варианта использования с некоторым другим вариантом использования, функциональность или поведение которого задействуется первым не всегда, а только при выполнении некоторых дополнительных условий.

Графически данное отношение обозначается как отношение зависимости в форме пунктирной линии с «V»-образной стрелкой, направленной от зависимого варианта использования к независимому варианту использования и соединенной с ним в так называемой *точке расширения (extension point)* (рис. 6). Зависимый вариант использования часто называют также *расширяющим*, а независимый – *базовым* или *расширяемым* вариантом использования. При этом линия со стрелкой обязательно помечается ключевым словом <<extend>>, записанным в форме стереотипа.

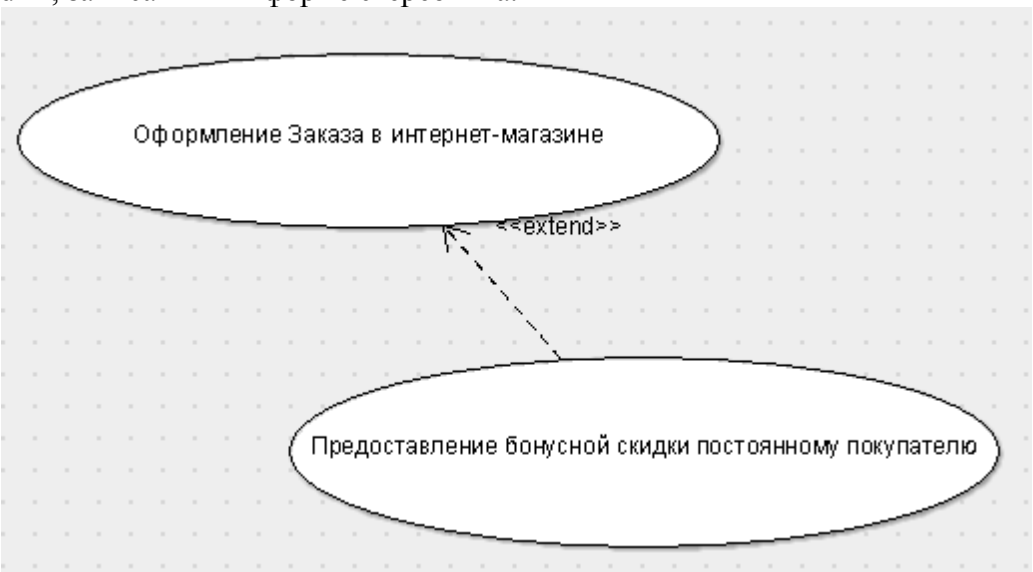


Рис. 6. Пример графического изображения отношения расширения между вариантами Использования.

3. *Отношение обобщения (generalization)* предназначено для спецификации того факта, что один элемент модели является специальным или частным случаем другого элемента модели.

Графически отношение обобщения обозначается сплошной линией со стрелкой в форме не закрашенного треугольника, которая указывает на общий элемент модели.

В изображенном на рис. 7 фрагменте диаграммы вариантов использования отношение обобщения специфицирует то, что вариант использования «Оплата товара по кредитной карточке» является специальным случаем варианта использования «Оплата выбранного в интернет-магазине товара». При этом вариант использования «Оплата выбранного в интернет-магазине товара», который является *потомком*, наследует все свойства поведения своего *родителя*, т.е. варианта использования «Оплата товара по кредитной карточке».

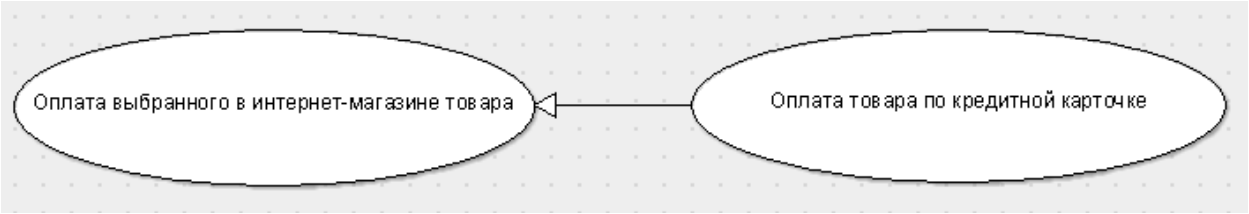


Рис. 7. Пример графического изображения отношения обобщения между вариантами использования

Пример графического изображения отношения обобщения между актерами показан на рис. 8.

Рис. 9. Пример графического изображения отношения обобщения между актерами

Модель вариантов использования

Требование (requirement) – желательное свойство, характеристика или условие, которым должна удовлетворять программная система в процессе своей эксплуатации.

Спецификация требований к проектируемой системе в форме диаграммы вариантов использования и дополнительных сценариев может представлять собой самостоятельную модель, которая в языке UML получила название ***модели вариантов использования***.

Содержание варианта использования может быть дополнительно специфицировано в форме пояснительного текста, который раскрывает смысл или семантику выполняемых действий при выполнении варианта использования. Такой пояснительный текст применительно к диаграммам вариантов использования получил специальное название текста-сценария или просто – сценария.

Сценарий (scenario) – специально написанный текст, который описывает поведение моделируемой системы в форме последовательности выполняемых действий актеров и самой системы.

Один из шаблонов для написания сценария отдельного варианта использования рассматривается в табл. 1.

Таблица 1. Шаблон для написания сценария отдельного варианта использования

Главный раздел	**Раздел «Типичный ход событий»**	**Раздел «Исключения»**	**Раздел «Примечания»**
Актеры Цель Краткое описание Тип Ссылка на другие варианты использования	Типичный ход событий, приводящий к успешному выполнению данного варианта использования	Исключение № 1	Примечание № 1
		Исключение № 2	Примечание № 2
		…	…
		Исключение № *n*	Примечание № *n*

Для удобства записи и чтения сценария часто используют модификацию данного шаблона, располагая разделы последовательно сверху вниз. При этом написание сценариев модели начинают с базовых или основных вариантов использования, после чего рассматриваются сценарии второстепенных или включаемых вариантов использования. Сценарии расширяющих вариантов использования обычно помещаются в разделе исключений базового варианта использования.

Пример построения модели вариантов использования

Построим модель вариантов использования для системы продажи товаров в интернет-магазине.

Посетитель интернет-магазина может просматривать список товаров интернет-магазина, помещать товар в виртуальную корзину и изменять содержимое этой корзины. Посетитель может стать покупателем, если он принимает решение об оформлении заказа на покупку выбранных им товаров.

Менеджер может изменять список товаров и специфицировать условия для предоставления бонусной скидки, а бухгалтер – принимать оплату за выбранный покупателем товар.

При оформлении заказа на покупку товара необходима регистрация покупателя.

При оформлении заказа постоянному покупателю может быть предоставлена специальная бонусная скидка.

В рамках рассматриваемой системы продажи товаров в интернет-магазине возможна оплата выбранных покупателем товаров, как наличными, так и по кредитной карточке.

Выделение актеров и вариантов использования

В соответствии с определением терминов «актер» и «вариант использования», для данной предметной области актерами будут люди, участвующие в работе интернет-магазина, а вариантами использования те действия, которые они выполняют в интернет-магазине.

1. В качестве основного актера описываемой системы можно рассматривать актера «Посетитель интернет-магазина». В соответствии с теми действиями, которые может выполнять посетитель интернет-магазина, можно выделить два варианта использования, свзанных с актером «Посетитель интернет-магазина» отношением ассоциации: «Просмотр списка товаров» и «Изменение содержания корзины».

2. Т.к. посетитель интернет-магазина может стать покупателем, то можно выделить актера «Покупатель», связанного отношением обобщения с актером «Посетитель интернет-магазина», при этом в качестве родителя будет выступать актер «Посетитель интернет-магазина», а в качестве потомка «Покупатель», т.к. последний наследует все свойства поведения посетителя интернет-магазина, но и обладает собственными. Вариантами использования, связанными отношением ассоциации с актером «Покупатель», будут «Оформление заказа на покупку товаров» и «Оплата выбранного товара».

3. В качестве других актеров рассматриваемой системы могут выступать «Менеджер» и «Бухгалтер». При этом «Менеджер» будет связан отношением ассоциации с вари-

антами использования «Изменение списка товаров» и «Предоставление бонусной скидки», а бухгалтер – с вариантом использования «Оплата выбранного товара».

4. Поскольку при оформлении заказа на покупку товара необходима регистрация покупателя, и эта функциональность выполняется всегда, она может быть выделена в отдельный вариант использования «Регистрация покупателя», который будет связан с базовым отношением включения. С другой стороны, при оформлении заказа постоянному покупателю может быть предоставлена специальная бонусная скидка. Это требование может быть также представлено в качестве отдельного варианта использования «Предоставление бонусной скидки», который будет связан с базовым отношением расширения.

5. Дальнейшая детализация модели может быть выполнена на основе установления дополнительного отношения обобщения для варианта использования «Оплата выбранного товара». Если в рамках рассматриваемой системы продажи товаров в интернет-магазине возможна оплата выбранных покупателем товаров, как наличными, так и по кредитной карточке, то в этом случае диаграмма может быть дополнена соответствующими вариантами использования. При этом варианты использования «Оплата товара наличными» и «Оплата товара по кредитной карточке» будут связаны с вариантом использования «Оплата выбранного товара» отношением обобщения.

В результате разрабатываемая диаграмма должна содержать 9 вариантов использования и 4-х актеров, между которыми установлены соответствующие отношения ассоциации, включения, расширения и

Создание диаграммы вариантов использования с помощью ArgoUML

ArgoUML – это бесплатная программа, предназначенная для моделирования с помощью UML.

Порядок построения диаграммы вариантов использования с помощью программы ArgoUML таков:

1. Запустите ArgoUML. Появится окно, изображенное на рис. 10.

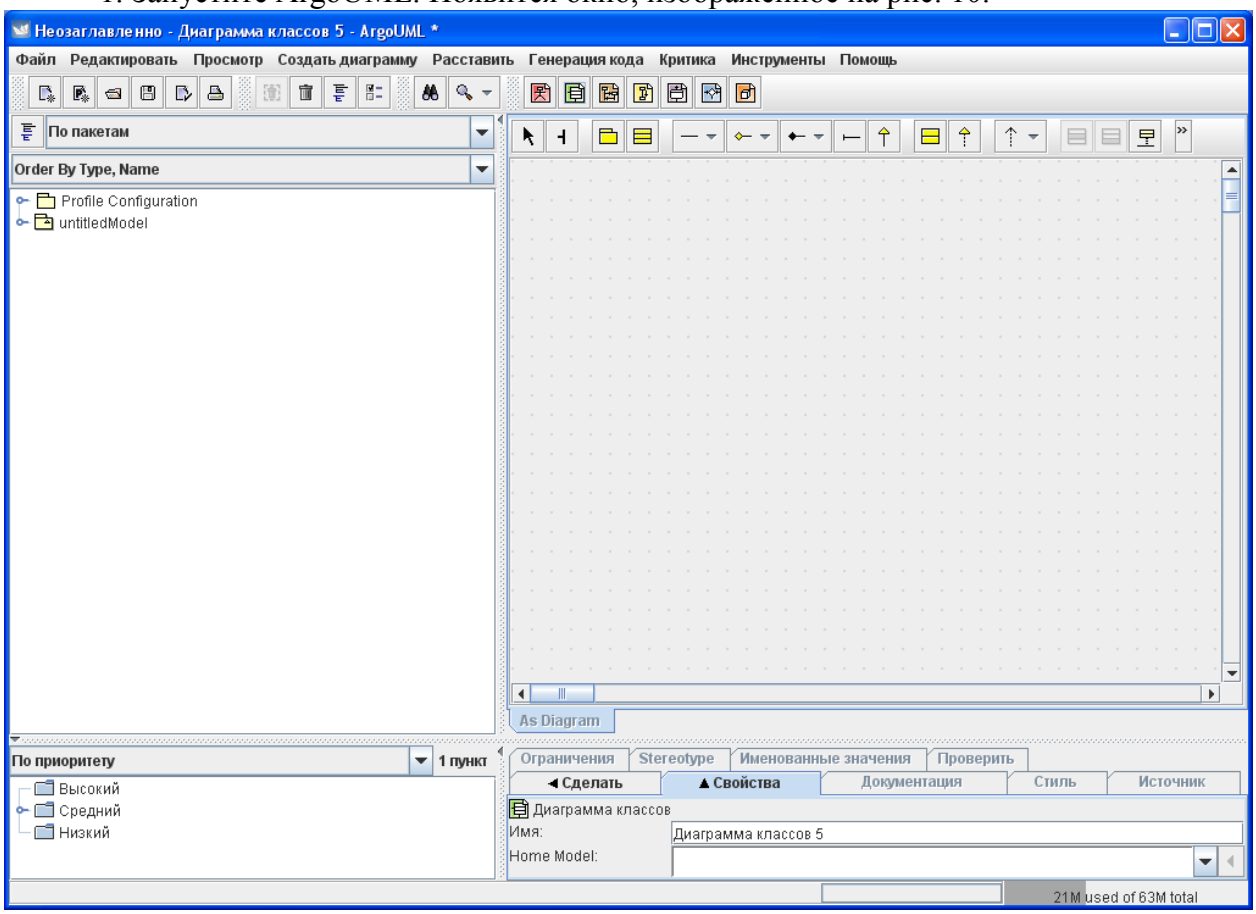


Рис. 10. ArgoUML.

2. Для того, чтобы создать диаграмму вариантов использования, на панели инструментов выберите пиктограмму *Диаграмма Вариантов использования* ( ).

3. Создадим актера. Выберем пиктограмму *Новый актер* ( ). Затем левой кнопкой мыши щелкнем по области, в которой будет изображена диаграмма. На ней появится новый актер (см. рис. 11).

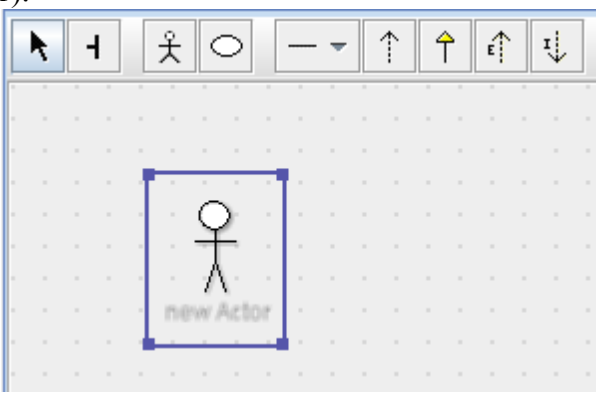


Рис. 11. Новый актер.

Установим для актера имя. Это можно сделать во вкладке *Свойства* (см. рис. 12).

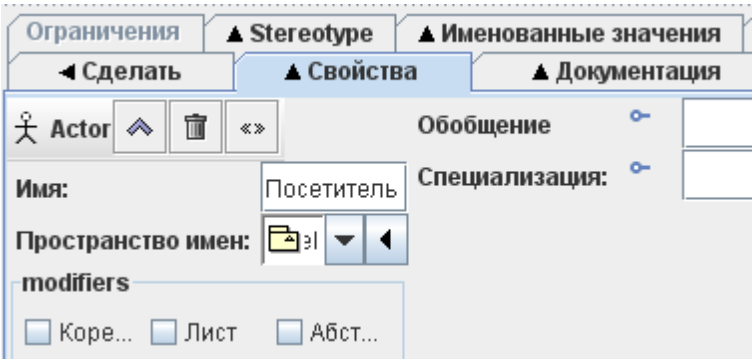


Рис. 12. Установка имени актера

Результат показан на рис. 13.

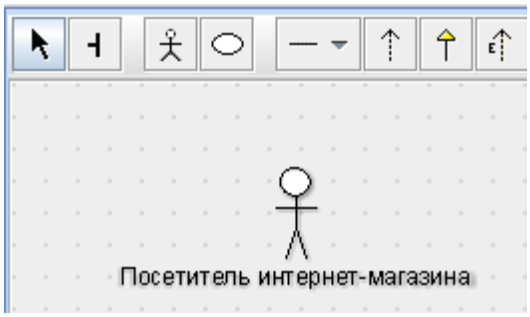


Рис. 13. Актер «Посетитель интернет-магазина».

4. Созданию вариант использования. Выберем пиктограмму *Новый вариант использования* ( ). Затем левой кнопкой мыши щелкнем по области, в которой будет изображена диаграмма. На ней появится новый вариант использования (см. рис. 14).

Установим для варианта использования имя. Это можно сделать во вкладке *Свойства* (см. рис. 15).

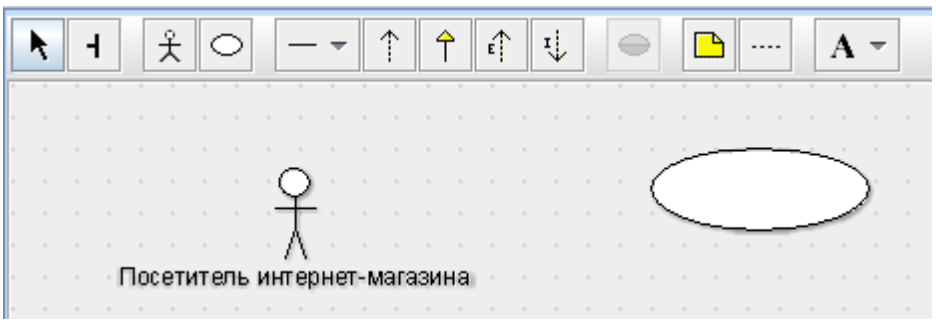


Рис. 14. Новый вариант использования.

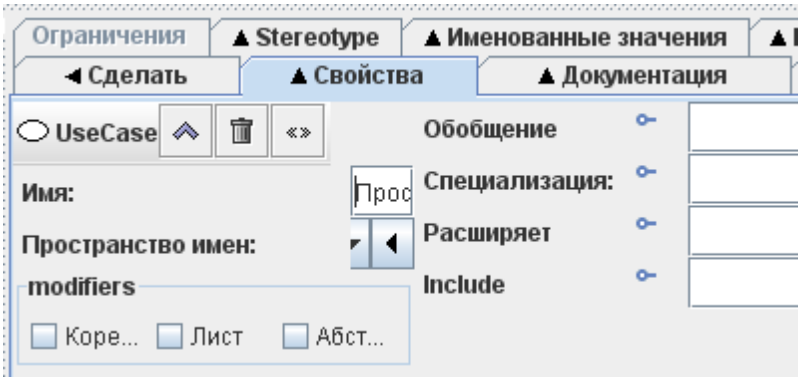


Рис. 15. Установка имени варианта использования

Рис. 16. Вариант использования «Просмотр списка товаров»

5. Установим отношение ассоциации между созданными актером и вариантом использования. Выберем пиктограмму *Новая ассоциация*( ). Проведем ее с зажатой левой кнопкой мыши от актера к варианту использования. Результат изображен на рис. 17.

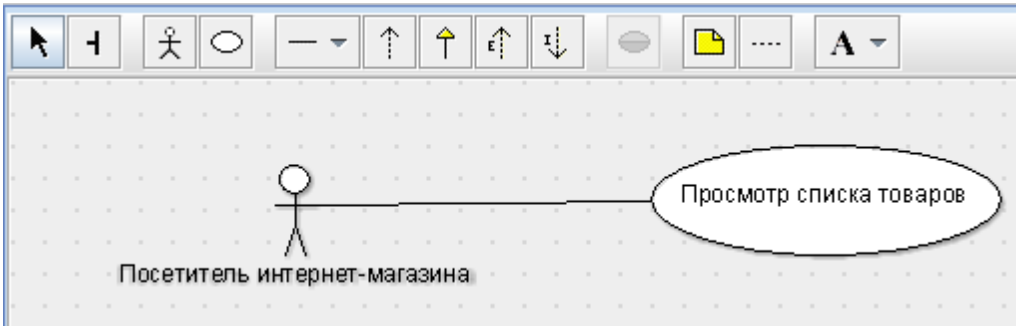


Рис. 17. Отношение ассоциации между актером и вариантом использования.

6. Аналогично пунктам № 2-5 создайте остальных актеров, варианты использования и установите отношения между ними. Результат показан на рисунке 18.

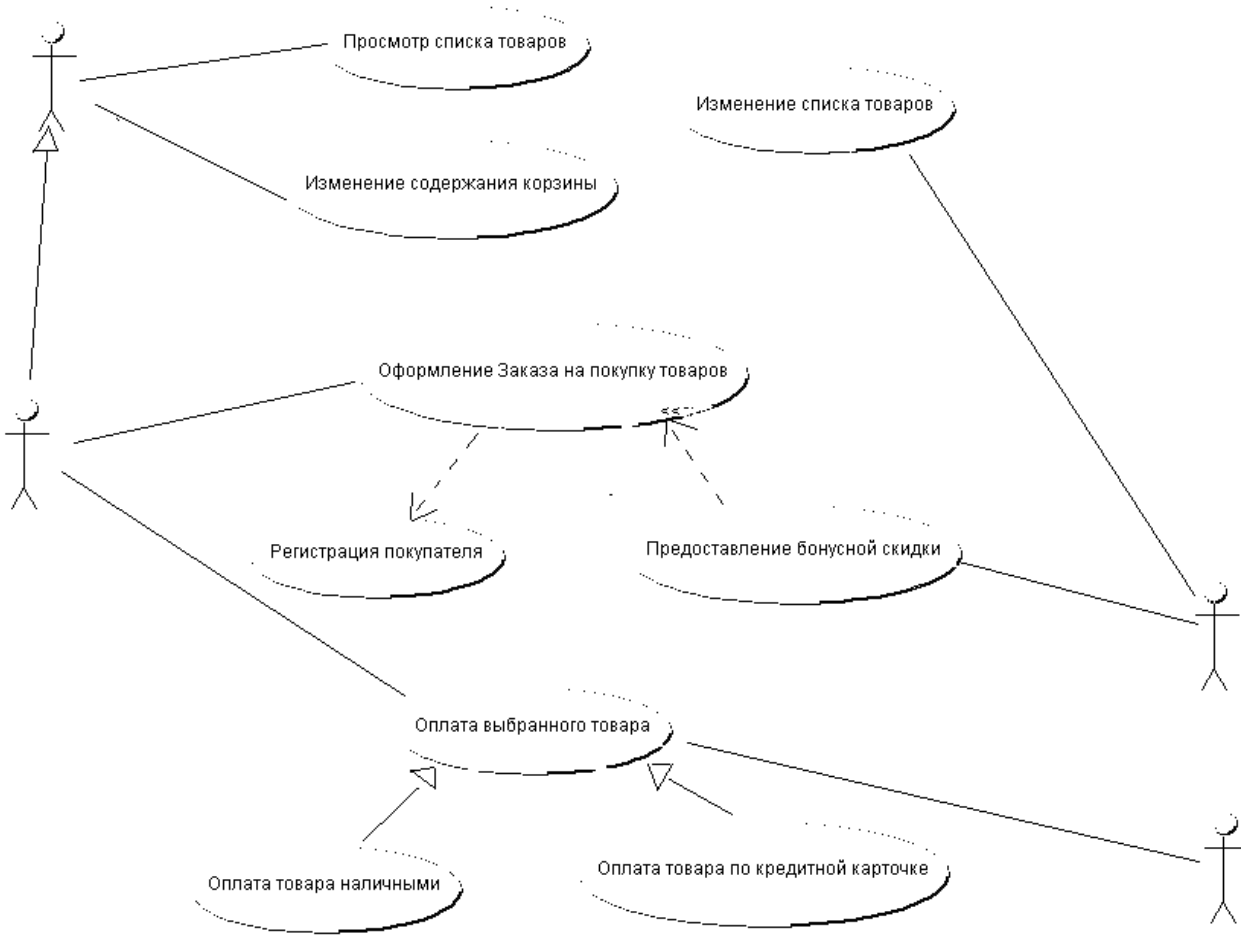


Рис. 18. Диаграмма вариантов использования для системы продажи товаров в интернет-магазине

7. Заключим созданные варианты использования в субъект. Для этого выберем пиктограмму *Прямоугольник* (). Поместим прямоугольник на область диаграммы. Изменим размеры прямоугольника так, чтобы он охватывал все варианты использования. Также у прямоугольника для свойства *Заполнить* на вкладке *Стиль* установим значение *Не заполнять* (см. рис. 19).

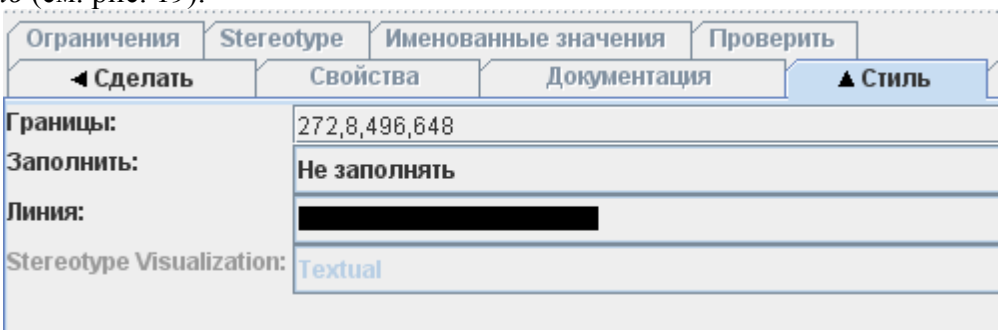


Рис. 19. Установка значения свойства для прямоугольника.

Чтобы создать имя субъекта, выберем пиктограмму *Текст* ( ). Поместим текст на область диаграммы. Во вкладке *Стиль* для раздела *Линия* установим значение *Не линия* (см. рис. 20).

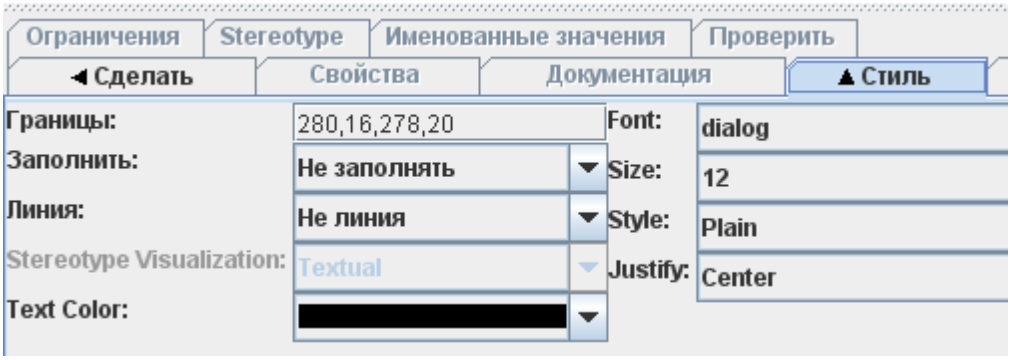


Рис. 20. Установка значения свойства для текста.

В итоге должна получиться диаграмма, изображенная на рис. 21.

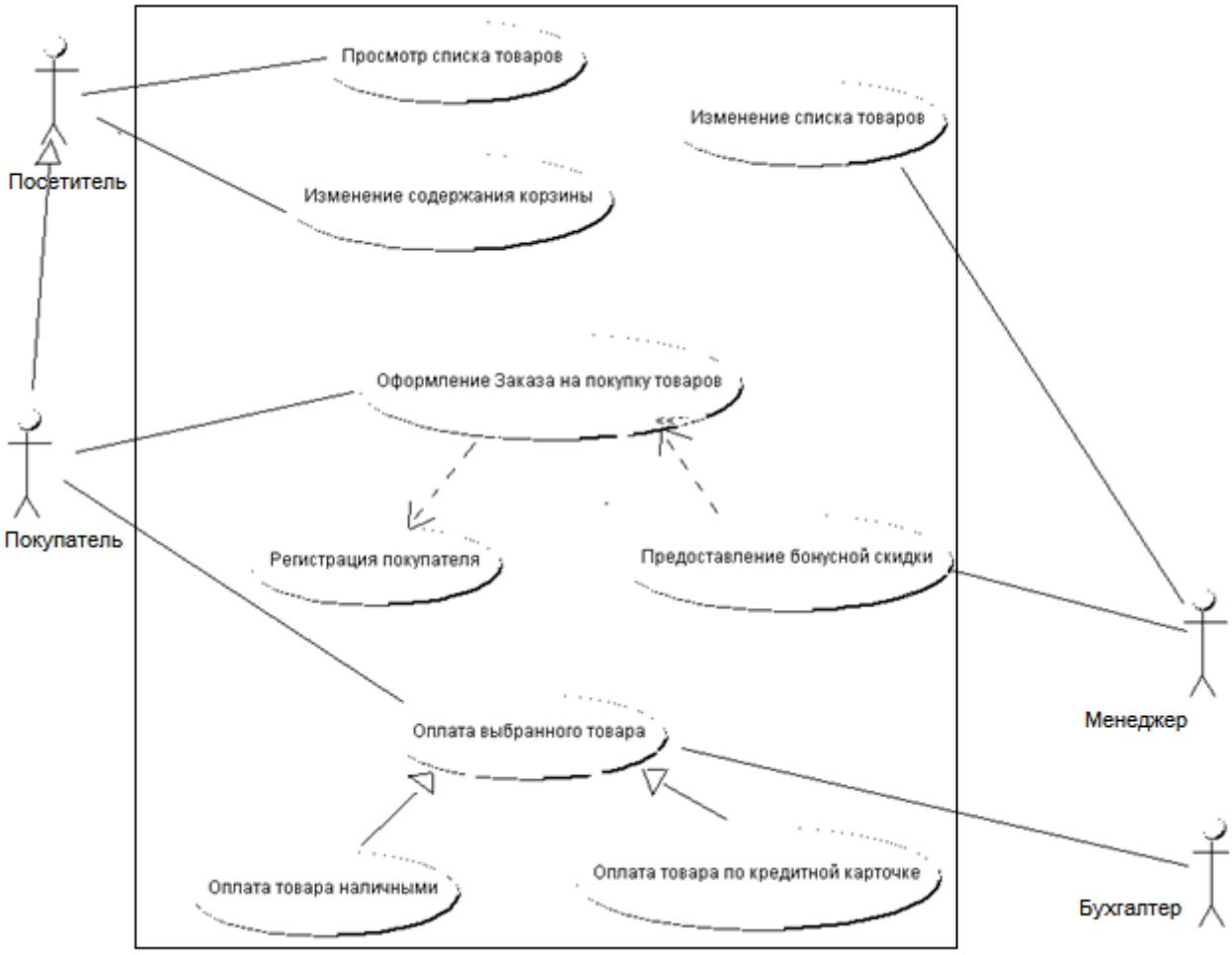


Рис. 21. Диаграмма вариантов использования для системы продажи товаров в интернет-магазине.

8. Чтобы сохранить диаграмму в графическом файле, выберем пункт меню *Файл->Сохранить диаграммы как графику*.

Написание текстового сценария

Дополним разработанную диаграмму вариантов использования текстовым сценарием, написанным на основе предложенного шаблона.

В качестве примера выберем вариант использования «Просмотр списка товаров», для которого соответствующий сценарий удобно представить в виде двух отдельных таблиц (таблица 2 и таблица 3), каждая из которых описывает отдельный раздел шаблона.

Таблица 2. Главный раздел сценария выполнения варианта использования «Просмотр списка товаров»

Вариант использования	**Просмотр списка товаров**
Актеры	Посетитель интернет-магазина
Цель	Получение требуемой информации о товарах, представленных в интернет-магазине
Краткое описание	Посетитель интернет-магазина просматривает информацию о товарах. Система обеспечивает доступ к любому товару и удобную навигацию по различным категориям товаров
Тип	Базовый
Ссылки на другие варианты использования	Отсутствуют

Таблица 3. Раздел типичного хода событий сценария выполнения варианта использования «Просмотр списка товаров»

Действия актеров	**Отклик системы**
Посетитель загружает исходную страницу интернет-магазина в браузер	Система отображает исходную страницу интернет-магазина
Посетитель интернет-магазина выбирает категорию интересуемых товаров	Система отображает информацию о выбранной категории товаров
Посетитель интернет-магазина выбирает интересуемый товар	Система отображает общую информацию о выбранном товаре
Посетитель интернет-магазина выбирает просмотр детальной информации об интересуемом товаре	Система отображает детальную информацию о выбранном товаре
Посетитель интернет-магазина может пожелать вернуться на исходную страницу интернет-магазина	Система отображает исходную страницу интернет-магазина

Поскольку для рассматриваемого варианта использования отсутствуют расширяющие варианты использования, секции исключений останутся пустыми.

Задание к работе

Выберите предметную область и постройте для этой предметной области модель вариантов использования, т.е. диаграмму вариантов использования и текстовые сценарии.

Содержание отчета

1) Титульный лист.

2) Диаграмма вариантов использования.

3) Текстовые сценарии.

Контрольные вопросы и задачи

1. Для чего используется язык UML?

2. Для достижения каких целей предназначена диаграмма вариантов использования?

3. Что такое вариант использования? Приведите примеры вариантов использования.

4. Что такое актер? Приведите примеры актеров.

5. Что такое субъект?

6. Что такое комментарий?

7. Что такое отношение?

8. Что такое отношение ассоциации?

9. В чем отличие направленной ассоциации от актера к варианту использования и от варианта использования к актеру?

10. Что такое отношение зависимости?

11. Что такое отношение включения?

10. Разработка и создание UML диаграмм.

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 ARGOUML (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.

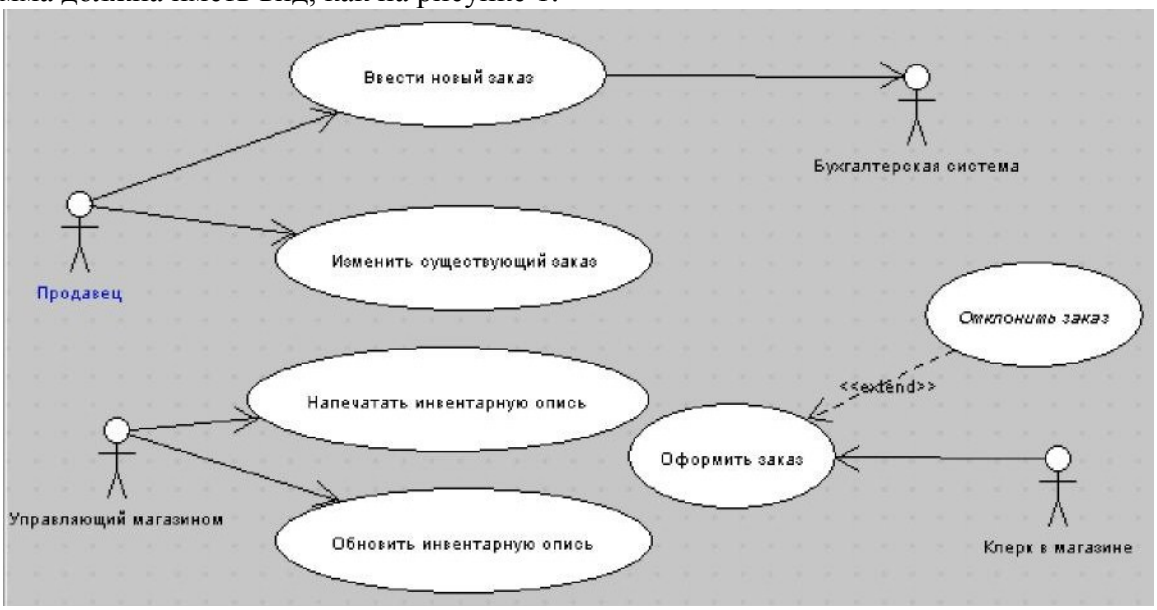


Рис. 1 – Диаграмма вариантов использования для системы обработки заказов

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).

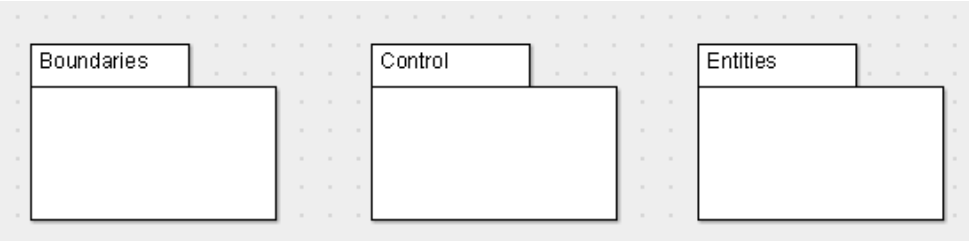


Рис. 2 – Главная диаграмма классов системы обработки заказов

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.

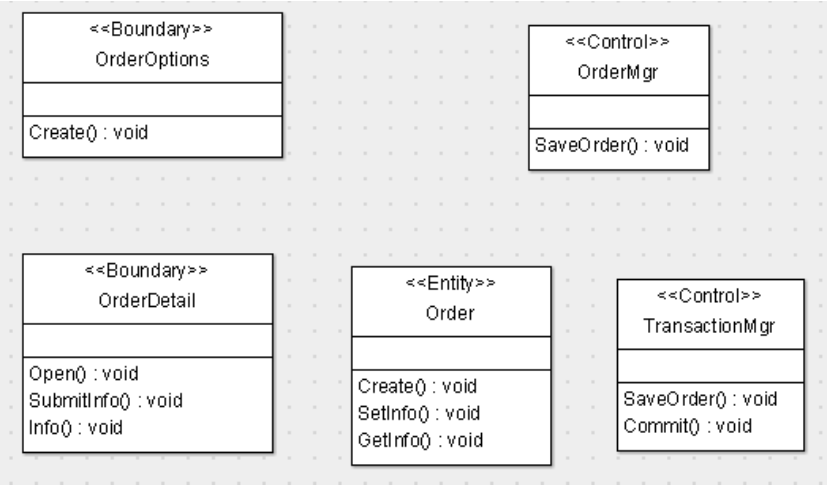


Рис. 3 – Стереотипы классов для варианта использования Ввести новый заказ

Задание 8. Объединение классов в пакеты

1) Перетащите в браузере 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 рефлесии. 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).

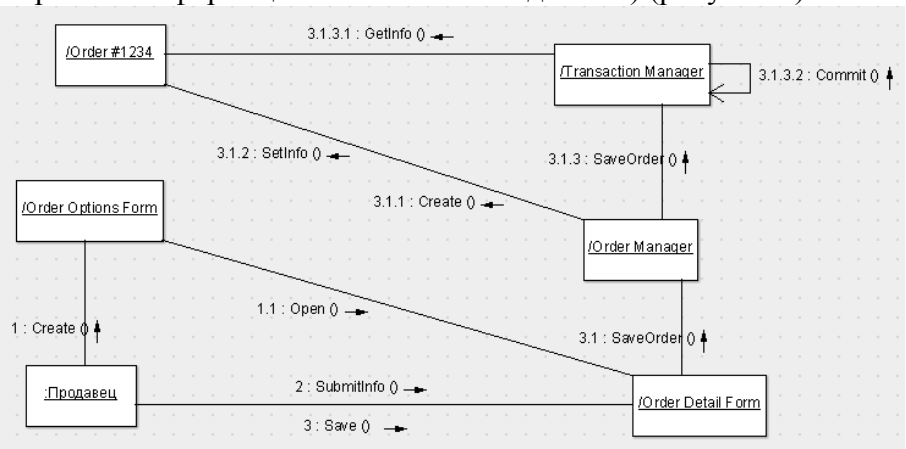


Рис. 4 – Диаграмма коопераций

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).

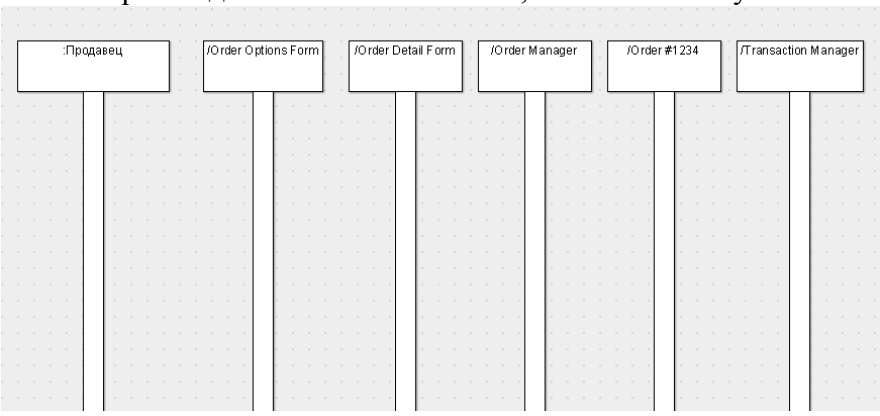


Рис. 5 – Диаграмма последовательности

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.

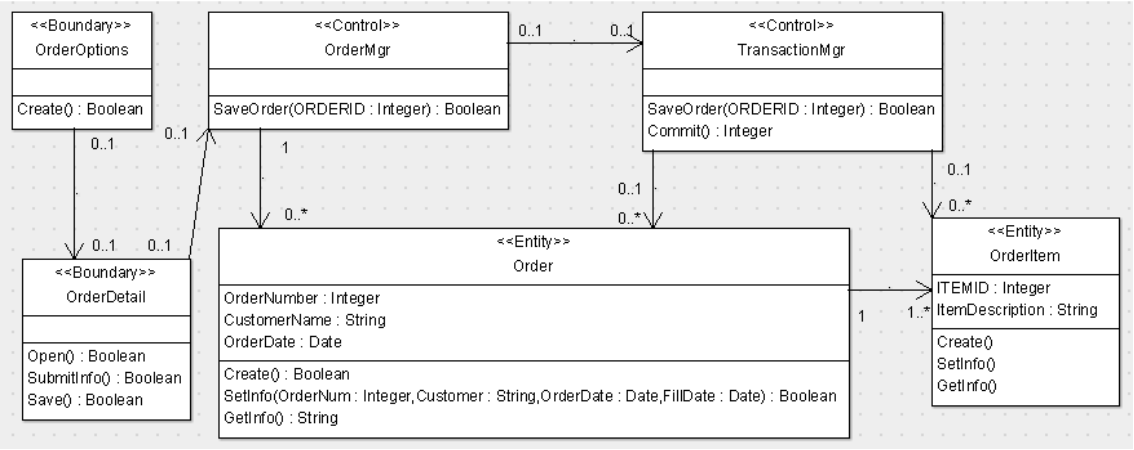


Рис. 6 – Диаграмма классов Add New Order

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 состояние Заполнены (Выполнено), открывая окно его свойств.

5) В поле Имя введите фразу Добавить Порядок пункта (Добавить в заказ на новое место).

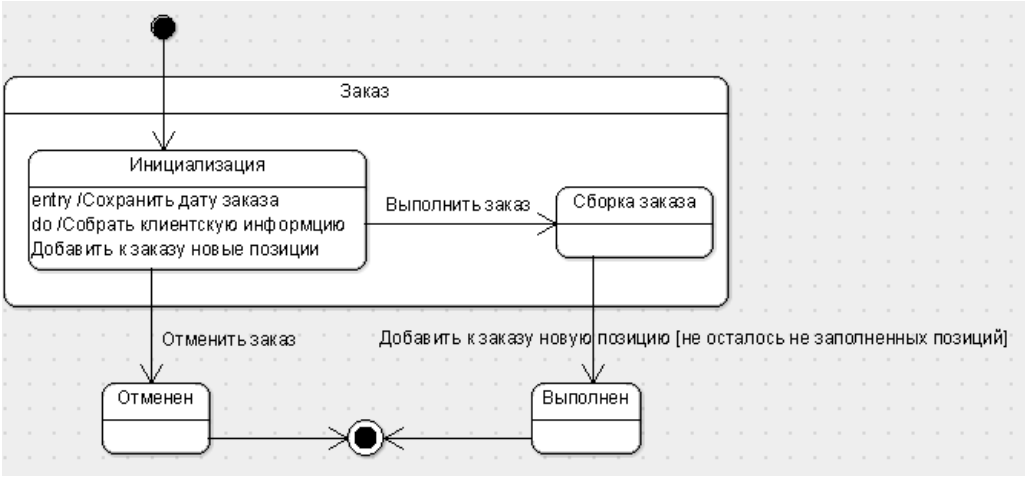


Рис. 7 – Диаграмма состояний

Задание 31. Добавление компонентов и изображение зависимостей

1) Выполните команду Создать диаграмму → Диаграмма развертывания.
2) Нажмите на название диаграммы в браузере. Введити 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.

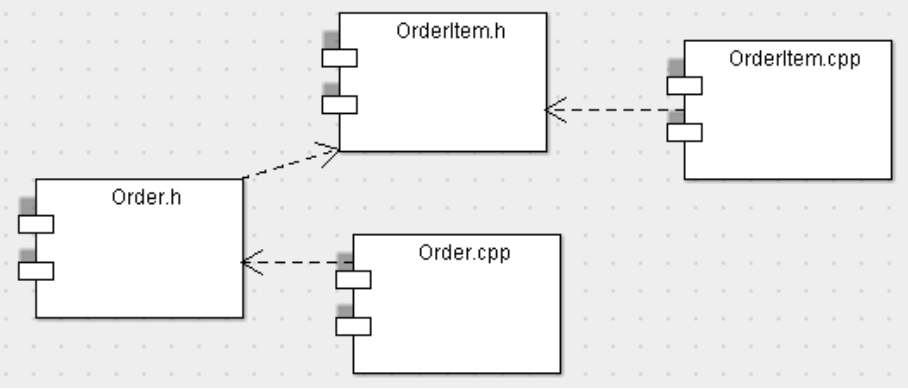


Рис. 8 – Диаграмма компонентов Entities

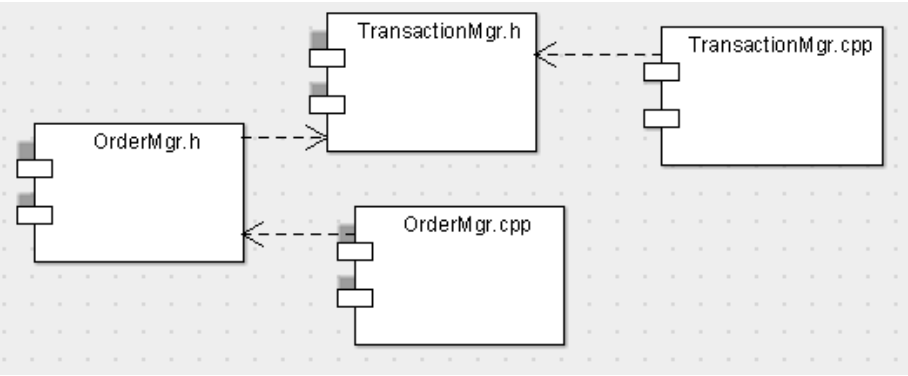


Рис. 9 – Диаграмма компонентов Control

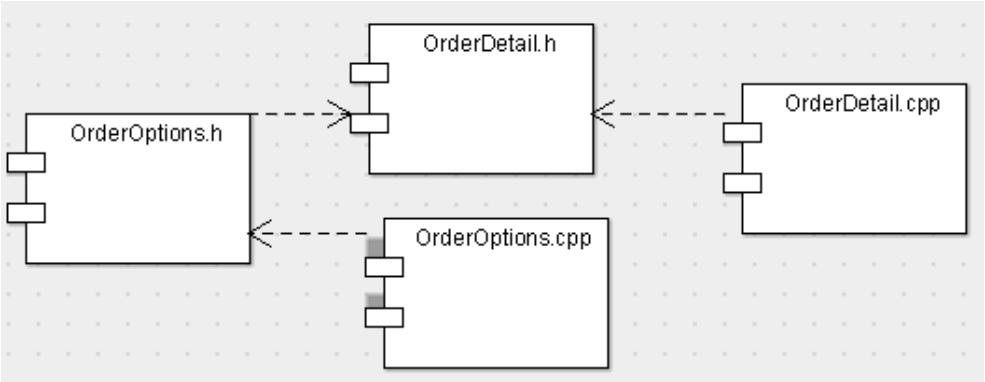


Рис. 10 – Диаграмма компонентов Boundaries

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, OrderOtions.s, 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.

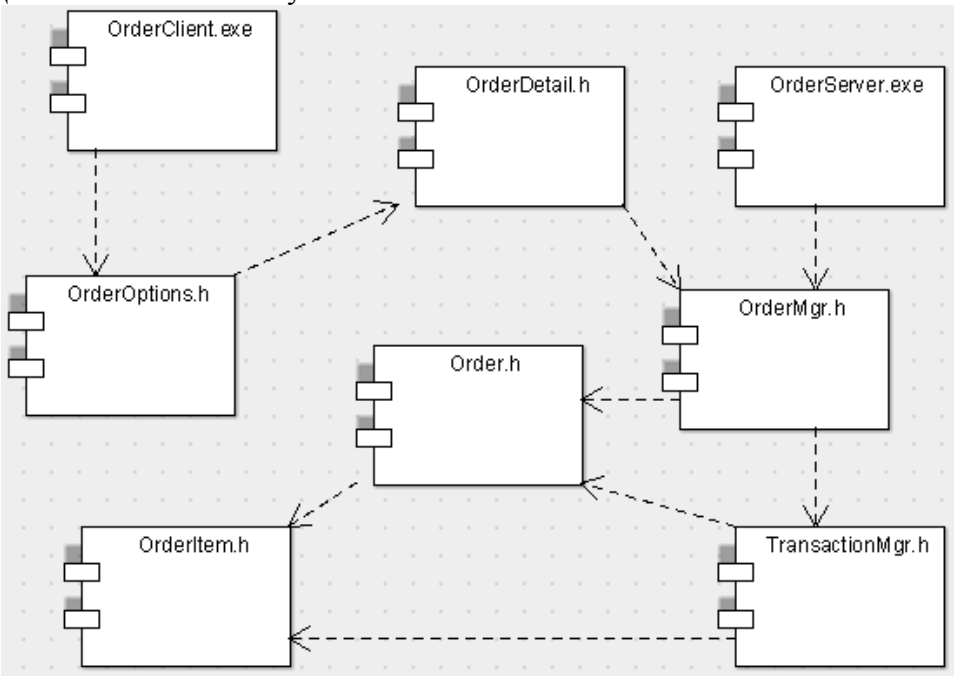


Рис. 12 – Диаграмма компонентов системы

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.

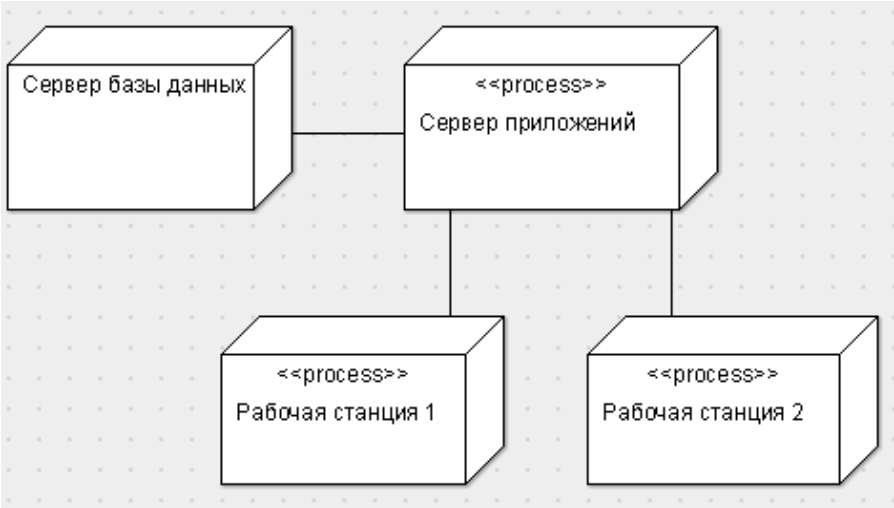


Рис. 13 – Диаграмма размещений

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.

11. Анализ требований к системе, планирование проекта.

Цели работы: 1. Рассмотреть этап анализа требований к системе. Познакомиться с актерами, прецедентами и пользовательскими историями.

2. Познакомиться с диаграммами UML, относящимися к данному виду деятельности.

3. Познакомиться с программным обеспечением для создания UML диаграмм.

4. Познакомиться с принципами экстремального планирования.

Приведем пример постановки задачи для небольшой информационной системы, предназначенной для кредитного отдела банка (или другой кредитной организации).

Краткая характеристика системы (первая постановка задачи от заказчика):

программное обеспечение предназначено для расчета платежей по потребительскому кредиту (и другим видам кредитов тоже) и в перспективе – для заключения и ведения кредитных договоров.

Первоначально планируется просто выполнять расчет среднемесячного платежа по фиксированной процентной ставке. Затем, возможно, потребуется рассчитывать помесячные выплаты (для тех клиентов, которые захотят платить неравными долями). В перспек-

тиве программа должна будет брать данные о процентах из какого-то внешнего источника. На первых порах программа будет использоваться только сотрудниками кредитного отдела, а потом, возможно, и клиентами удаленно (например, с сайта компании). Также, возможно, программа в будущем будет использоваться и при заключении договоров, то есть будет допускать ввод информации о клиенте, сохранять информацию о параметрах его кредита, и даже может быть – о его платежах и истории выплат, выполнять перерасчет его долга при отклонениях от графика погашения, начислять штрафы и прочее. Еще в более дальней перспективе программа может быть связана с межбанковской системой кредитных историй и дополнена алгоритмом проверки клиента. Альтернативный вариант развития системы – ее интеграция с АБС.

Что касается интерфейса программы, то он будет графическим, хотя на первых порах она должна просто считать! Затем, возможно, потребуется Web интерфейс. Также и с сетевыми возможностями – пусть хотя бы программа будет работать на одном компьютере локально, а там, вполне вероятно, потребуется сделать ее сетевой, и даже, как упоминалось выше, разместить аналогичный модуль (или просто форму) в Интернет. Не исключен в дальней перспективе и мобильный клиент.

Предварительный анализ.

Пробуем выделить всех возможных заинтересованных лиц, а также прецеденты. Итак, прежде всего заинтересованными лицами для этой системы (назовем ее пока CreditLite) являются :

– собственно работник кредитного отдела, работающий с клиентом (Менеджер);

– клиент;

– в будущем – удаленный клиент;

– еще в более дальнем будущем – мобильный клиент;

– внешняя база данных, где хранятся проценты (или как далее – АБС);

– возможно – АБС (как источник или приемник данных, либо – как внешняя система);

– возможно – межбанковская система (в будущем);

– любой интересующийся (потенциальный клиент);

– клиент, заключивший договор и выплачивающий кредит и пр.

В принципе, если проанализировать этот список, то ряд заинтересованных лиц являются (по крайней мере на первых порах) просто более специфическими ролями базовых заинтересованных лиц:

– менеджера;

– клиента;

– внешней информационной системы.

Обратимся к прецедентам. Из неформального описания функций системы можно выделить следующие прецеденты:

П1 Расчет платежа по кредиту (за определенный период).

П1.1 Расчет среднемесячного платежа по кредиту

П1.2 Расчет помесячных платежей

П2 Заключение договора о кредите

П3 Перерасчет кредита (суммы основного долга, процентов и пр.) после погашения его части с учетом графика

П4 Начисление штрафов за просрочку

П5 Перерасчет кредита с учетом штрафов

П6 Загрузка кредитной истории

П7 Выгрузка кредитной истории

П8 Анализ анкеты клиента

П9 Анализ анкеты клиента с учетом кредитных историй

П10 Выгрузка информации о договоре в АБС и пр.

Остальное (например, превращение системы в АБС) пока находится даже за

пределами фантазий самого заказчика.

Можно составить краткое (сжатое) описание каждого прецедента (это могут сделать сами студенты:

П1 Расчет платежа по кредиту. Системе сообщается месячная (либо годовая) процентная ставка, сумма кредита и количество месяцев (срок), на который он берется.

Система рассчитывает сумму начисляемых за все время пользования кредитом процентов с учетом ежемесячного погашения кредита равными долями и общую сумму выплат. (Для этого прецедента интересно составить диаграмму деятельности, то есть фактически – алгоритм!)

П1.1 Расчет среднемесячного платежа по кредиту – практически это то же самое, но система делит общую сумму выплат на количество месяцев (можно не выделять в качестве отдельного прецедента).

П1.2 Расчет помесячных платежей – выполняется почти такой же расчет, как и в предыдущем прецеденте, но – неравными долями (клиент выплачивает сумму кредита равными долями, а процент за кредит уменьшается от месяца к месяцу) (Последние два прецедента не очень похожи на настоящие прецеденты, они почти не имеют собственного значения и являются подпрецедентами, либо же вообще их нужно объединить с первым).

П2 Заключение договора о кредите – Клиент обращается в кр. Отдел и сообщает сумму и срок кредита. Менеджер производит расчет, при этом задействуются прецеденты

П1, П1.1, П1.2. Если Клиента устраивают условия кредита, он заполняет анкету. Менеджер анализирует ее (либо в будущем активируются прецеденты П6 и П8-П9), принимает решение о выдаче кредита (если да), затем вводит параметры договора в систему (возможно, при этом задействуется прецедент П10). - Для этого прецедента рекомендуется рассмотреть диаграмму активности, диаграмму взаимодействия, а сначала – рассмотреть подробное описание прецедента с альтернативными ветвями и пр. Это “настоящий” прецедент, который характеризуется явным значимым результатом и для Менеджера, и для Клиента, и для Кредитной организации и т.д. ...

Теперь проанализируем (вместе с заказчиком) приоритеты прецедентов, риски и сроки для того, чтобы составить первоначальный план версий. Трудозатраты на каждое пожелание пользователя (прецедент, историю) оценим в **идеальных неделях**.

Идеальная неделя (день) - условная единица измерения времени, а вернее сказать – сложности задачи при планировании в XP. Под одним идеальным днем понимается 8-часовой рабочий день, в течение которого разработчик занимается только решением конкретной задачи (для которой оценивается сложность), не отвлекается на посторонние дела, он чувствует себя хорошо, работа “идет”, одним словом, разработчик работает со своей нормальной производительностью, близкой к максимальной. Идеальная неделя – это 5 рабочих (идеальных) дней. Одна итерация занимает от 1 до 4 недель (в зависимости от проекта, условий, команды, но в одном проекте продолжительность итерации фиксированная). Одна версия включает обычно минимум 2 итерации.

Список прецедентов дополним существенными пожеланиями пользователя по поводу свойств системы, такими как сетевые возможности, Web-интерфейс, доступ через Интернет, мобильный доступ, чтение условий кредитования из внешнего источника и пр. Очередность реализации пожеланий определяется совместно заказчиком и разработчиком исходя из приоритетов задач, их объемов, рисков. Риск в данном случае – это некоторая угроза для проекта. К рискам можно отнести : неустойчивость работы системы (частые сбои, ошибки), некорректные реализации алгоритмов или некорректные алгоритмы, приводящие к ошибкам в расчетах, низкую производительность системы, технические трудности в реализации программ из-за отсутствия опыта работы с конкретным средством разработки, сторонней библиотекой, технологией, протоколом и пр.

В данном случае подобный вид первоначального плана версий объясняется прежде всего пожеланиями заказчика. Он определил, что его прежде всего интересует собственно

расчет платежей, затем ему зачем-то нужно обязательно иметь сетевую версию, а уже потом - перерасчет платежей (допустим, ему кажется, что в основном все клиенты будут платить во-время). Также почему-то заказчик предпочел сначала перенести проект в Интернет, а потом уже расширять его функциональность (возможно, он еще не решил, стоит ли вообще это делать). Ну, тем лучше для нас – это позволит нам сосредоточиться на важных технических деталях.

Пожелания клиента	**Оценка трудозатрат, в ид. неделях**	**Реализовать в итерации #**	**Реализовать в версии #**
Расчет платежа по кредиту (за определенный период) и среднемесячного платежа.	1	1	1
Расчет помесячных платежей	1	2	1
Заключение договора о кредите (просто фиксация данных о клиенте и параметрах договора, без анализа анкеты и кредитной истории)	1	5	2
Перерасчет кредита	2	4	2
Перерасчет кредита с учетом штрафов	1	5	2
Загрузка кредитной истории	2		
Выгрузка кредитной истории	2		
Анализ анкеты клиента	2		
Анализ анкеты клиента с учетом кредитных историй	2		
Заключение договора с анализом анкеты и кредитной истории	1		
Выгрузка информации о договоре в АБС	2		
Чтение условий кредитования из БД	1	2	1
Сетевая версия (для ЛВС)	2	3	2
Веб-интерфейс	2	6	3
Доступ через Интернет	2	7	3
Мобильный доступ	3		

Заметим, что прецеденты немного изменены, в частности, прецедент заключения договоров разбит на два, причем вторая часть отнесена на неопределенный срок. Некоторые пожелания вообще не фигурируют в обозримом списке итераций. Это связано с теми оценками времени, которые сделал разработчик для каждой задачи. Поскольку итерация имеет фиксированный размер (2 идеальные недели в данном случае), часть задач просто не уместилась в обозримый план из 14 идеальных недель (это примерно 3 идеальных месяца).

Заметим также, что реально на реализацию проекта даже в таком виде уйдет заведомо больше 3 месяцев, так как недели и дни не могут быть все идеальными. Так что можно смело умножить 3 месяца на 2 и получим полгода! Это не очень устраивает заказчика, так как через полгода он хотел бы уже иметь все задачи решенными, но ему приходится согласиться с оценками разработчика (хотя в душе он надеется на лучшее). Разработчик тоже надеется на некоторое ускорение работы по мере хода проекта (поэтому, например, он выделил на первую задачу трудоемкостью в 1 неделю целую итерацию, а затем стал планировать по две таких задачи в итерацию). Оценки трудоемкости выбираются разработчиком на основании его предыдущего опыта и представлений о задаче на момент составления плана. Однако в будущем его оценки изменятся в зависимости от той информации, которую он будет получать в ходе проекта (как от заказчика, так и от самого проекта, то есть как бы от себя)!

Медленная скорость работы разработчика при реализации первой версии объясняется еще и тем, что он :

– хочет сосредоточиться на создании корректного ключевого алгоритма и его реализации;

– хочет создать инфраструктуру проекта (папки, документы, диаграммы, проект для конкретного средства разработки и пр.);

– хочет освоить новые для него технологии разработки (XP, модульное тестирование, рефакторинг, шаблоны, диаграммы UML и пр.);

– вынужден осваивать не очень знакомые ему особенности бесплатной библиотеки GUI и особенности работы с бесплатным средством разработки, так как это одно из нефункциональных требований заказчика.

Несмотря на эти веские причины, конечно, трудно представить, то в реальности программисту на реализацию двух не очень сложных алгоритмов потребуется не два дня, а две недели (то есть 10 дней). Но не стоит забывать, что речь идет об учебном проекте! И, кроме того, еще раз вспомним о понятии идеальных дней (в реальности их приходится умножать не на 2, а на 3 или даже 4; лишь бы не на 10 и не на бесконечность.

Остановимся на более подробном плане первой итерации.

Итак, на первую итерацию запланирована всего одна задача (прецедент):

Расчет платежа по кредиту (за определенный период) и среднемесячного платежа. Нужно расшифровать эту задачу и выяснить, что нужно сделать для ее реализации уже в масштабе идеальных дней. Допустим, что заказчик пока согласился, что результатом первой итерации будет программа, не имеющая графического интерфейса (то есть у нее интерфейс командной строки), она не будет использовать БД или другие источники данных, а будет использовать фиксированный процент, который сообщил заказчик. При этом такая программка уже будет что-то считать, а значит, будет полезна заказчику (то есть за нее можно уже и денег запросить. Что нужно сделать, чтобы создать такую программу? Ну, во-первых, нужно разработать алгоритм. Во-вторых, поскольку программист решил работать в рамках ООП, то нужно создать класс, инкапсулирующий этот алгоритм (или алгоритмы). В третьих, поскольку разработчик решил работать по канонам XP, ему нужно создать тесты для тестирования своего класса, причем двух видов : приемочные тесты для оценки правильности расчета (их он составляет с помощью заказчика, или вообще получает от заказчика) и модульные тесты для реализации самого класса и его рефакторинга.

Поскольку программист не очень знаком с этими видами деятельности, принятыми в XP (тестирование, рефакторинг, разработка через тестирование), то он пока решает запланировать каждый вид деятельности отдельно (хотя на самом деле эти виды деятельности выполняются последовательно – параллельно и составляют собственно процесс разработки классов в XP).

Кроме того, разработчик хочет отвести некоторое время на проектирование всей разработки (не очень большое, как принято в XP), хотя бы на архитектурное. Ну и, как упоминалось выше, ему нужно время на создание начальной инфраструктуры проекта, необходимых документов, диаграмм, других артефактов, и время на изучение средств разработки и библиотек.

Получаем следующий примерный план итерации (P – разработчик):

Расчет платежа по кредиту (за определенный период) и среднемесячного платежа	**Трудозатраты в идеальных днях**	**Ответственный**
	5 – 10 (5 в запасе)	**P**
Составление плана итерации	0,5	
Создание проекта и инфраструктуры	1	
Создание эскизного проекта архитектуры	0,5	

Изучение особенностей средства разработки	1	
Изучение оболочки для тестирования	1	
Создание приемочного теста	0,5	
Создание модульных тестов	1	
Разработка класса с алгоритмом расчета	2	
Разработка приложения	1	
Оценка скорости разработки, планирование (в том числе текущее)	0,5	
Итого:	**9**	

В общем случае, когда разработчик не один, последний столбец реально используется и позволяет примерно поровну распределить задачи.

В принципе, целесообразно также составлять текущие планы на каждый день. Там прописываются конкретные дела (создание артефактов, элементов интерфейса, таблиц и запросов к БД, классов и методов, или, как мы увидим в дальнейшем, необходимые модульные тесты). Это помогает следить за скоростью разработки и даже несколько повышать ее. В таком плане по итогам дня вычеркиваются выполненные пункты и можно оценить реальную скорость.

Порядок выполнения работы

I Анализ требований

1. Изучить представленный пример(ы) постановки задачи, прецедентов, требований, а также диаграммы UML

2. Создать диаграмму прецедентов и другие диаграммы из примеров в программных средствах ArgoUML и ObjectiF.

3. Создать собственный прецедент и собственные диаграммы по вариантам.

II Планирование

1. Изучить примеры планирования.

2. Создать собственный план по вариантам.

Задания для составления прецедентов, диаграмм, планов (типовые проекты)

Составить список прецедентов и актеров, расписать 1-2 прецедента подробнее, составить план версий и итераций. Например :

1. Интернет – магазин.
2. Информационная система для учета кадров на небольшом предприятии
3. Информационная система для расчета зарплаты на небольшом предприятии
4. Утилита мониторинга сети (для администратора сети)
5. Обучающая программа с функцией контроля знаний (это основная функция!) (фактически кусок этого варианта рассмотрен выше в п. 2.7)
6. Биллинговая система для учета интернет-трафика.
7. Веб - сервер
8. Почтовый сервер
9. Текстовый редактор (потом – текстовый процессор?!)
10. Программа моделирования цифровых схем
11. Информационная система складского учета (учет прихода и ухода товаров, категории, поиск и пр.)
12. АРМ менеджера компьютерной фирмы (не бухгалтерия !). - Быстрая прикидка конфигураций, выписка счетов, отслеживание счетов их оплат и пр.
13. АРМ начальника отдела техподдержки (учет ремонтов).
14. Графический редактор (векторная графика).
15. Графический редактор (растровая графика).

12. Диаграммы состояний.

Цель работы: получить навыки построения диаграмм состояний.

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

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

Состояние	Описание состояния

Приемы работы с диаграммами состояний в ArgoUML

Теоретические сведения

Диаграммы состояний используются для моделирования динамических аспектов системы.

По большей части под этим подразумевается моделирование поведения реактивных объектов. Реактивным называется объект, поведение которого лучше всего характеризуется его реакцией на события, произошедшие вне его собственного контекста. У реактивного объекта есть четко выраженный жизненный цикл, когда текущее поведение обусловлено прошлым. Диаграммы состояний можно присоединять к классам, прецедентам или системе в целом для визуализации, специфицирования, конструирования и документирования динамики отдельного объекта.

Диаграмма состояний (StateChart, StateMachine diagram) показывает автомат, фокусируя внимание на потоке управления от состояния к состоянию.

Автомат (State machine) - это описание последовательности состояний, через которые проходит объект на протяжении своего жизненного цикла, реагируя на события, - в том числе описание реакций на эти события.

Состояние (State) - это ситуация в жизни объекта, на протяжении которой он удовлетворяет некоторому условию, осуществляет определенную деятельность или ожидает какого-то события.

Событие (Event) - это спецификация существенного факта, который происходит во времени и пространстве. В контексте автоматов событие - это стимул, способный вызвать срабатывание перехода.

Переход (Transition) - это отношение между двумя состояниями, показывающее, что объект, находящийся в первом состоянии, должен выполнить некоторые действия и перейти во второе состояние, как только произойдет определенное событие и будут выполнены заданные условия.

Действие (Action) - это атомарное вычисление, которое приводит к смене состояния или возврату значения.

Диаграмма состояний изображается в виде графа с вершинами и ребрами.

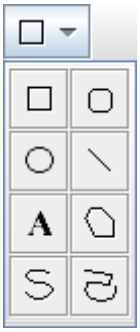
Панель инструментов

Для создания и редактирования диаграмм состояний (StateMachine) в ArgoUML предназначена панель инструментов, которая расположена горизонтально над окном диаграммы.

Табл.1. Панель инструментов для диаграмм состояний

Вид кнопки	Название кнопки	Назначение кнопки
	Select	Превращает курсор в стрелку указателя для того, чтобы можно было выделять и перемещать объекты
	Broom	Превращает курсор в «метлу», или «уровень», с помощью которого можно выравнивать объекты по линии
	Простое состояние	Добавление на диаграмму нового состояния
	Композитное состояние	Добавление на диаграмму нового композитного состояния
	New Transition	Добавление на диаграмму нового перехода
	Состояние синхронизации	Добавление на диаграмму символа синхронизации
	Состояние-ссылка на вложенный автомат	Добавление на диаграмму ссылки на автомат
	Состояние-заглушка	Добавление на диаграмму состояния-заглушки
	New Initial	Добавление на диаграмму стартового состояния
	Конечное состояние	Добавление на диаграмму конечного состояния
	New Junction	Добавление на диаграмму узла слияния
	New Choice	Добавление на диаграмму узла принятия решения
	New Fork	Добавление на диаграмму узла ветвления

Вид кнопки	Название кнопки	Назначение кнопки
	New Join	Добавление на диаграмму узла объединения
	New Shallow History	Добавление на диаграмму недавнего исторического состояния
	New Deep History	Добавление на диаграмму давнего исторического состояния
	Событие	Добавление на диаграмму события (вызова, изменения, сигнала, времени)
	New Guard	Добавление на диаграмму сторожевого условия.
	Действие	Добавление на диаграмму действия (вызова, создания, уничтожения, возврата, отправки, конечного состояния, не интерпретируемого, последовательности действий)
	New comment	Добавление к диаграмме примечания
	Comment link	Связывание примечания с объектом на диаграмме
	Инструменты	Стандартные инструменты (прямоугольник, скругленный прямоугольник, овал, линия, текст, многоугольник, кривая линия, чер-

Вид кнопки	Название кнопки	Назначение кнопки
		нила)

Создание новой диаграммы состояний

Для того чтобы создать новую диаграмму состояний в ArgoUML, необходимо кликнуть по кнопке панели инструментов, либо в окне браузера проекта кликнуть правой кнопкой по необходимой модели, раскрыть меню CreateDiagram и выбрать пункт Диаграмма состояний.

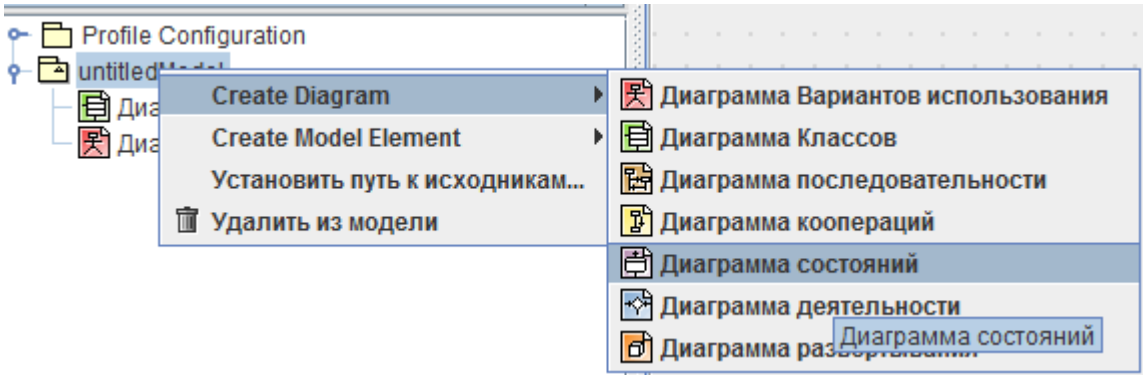


Рис.1. Создание диаграммы состояний.

Далее следует выделить только что созданную диаграмму и задать ей имя в окне свойств, которое расположено под окном диаграммы.

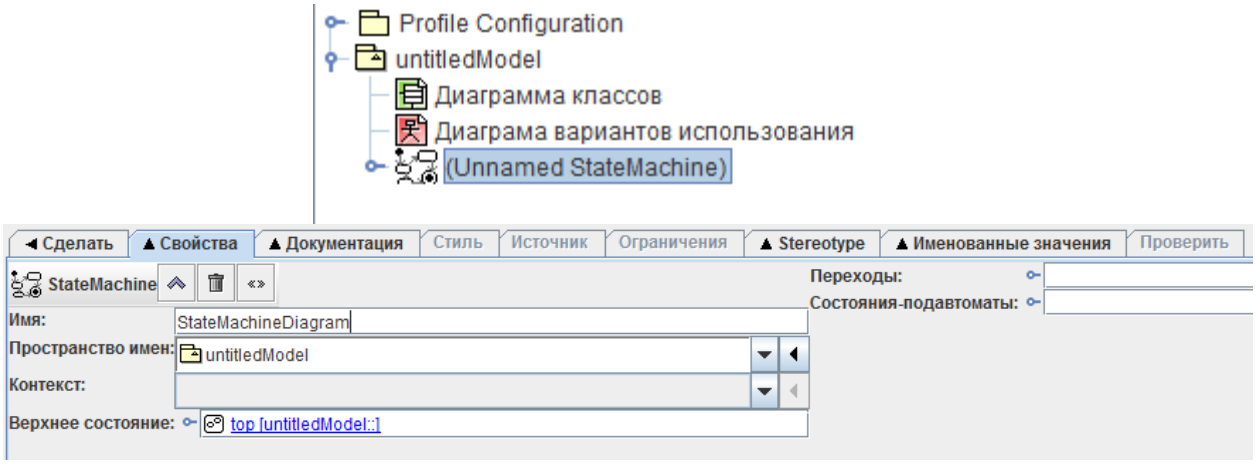


Рис.2. Свойства диаграммы состояний.

Добавление нового состояния

Для создания нового состояния нужно щелкнуть по кнопке *Простое состояние* на панели инструментов и затем по свободному месту диаграммы. При выделении любого элемента с помощью инструмента *Select* (при создании элемента он выделяется автоматически) под окном диаграммы открывается окно редактирования свойств элемента.

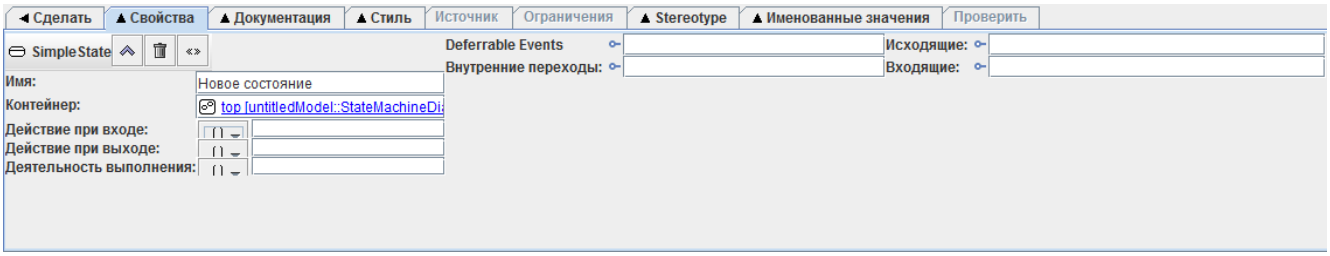
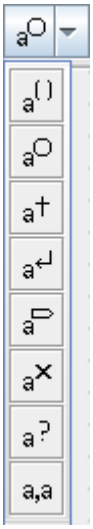


Рис.3. Свойства простого состояния.

Для каждого состояния можно определять действия (как в автомате Мура). В ArgoUML можно задать 3 действия для каждого состояния: действие при входе в состояние, действие при выходе из состояния и деятельность выполнения. Для того чтобы задать действие, нужно раскрыть выпадающее меню и выбрать тип действия. Всего в ArgoUML определены 8 типов действий:



- действие вызова;
- действие создания;
- действие уничтожения;
- действие возврата;
- действие отправки;
- действие конечного состояния;
- не интерпретируемое действие;
- последовательность действий.

Для выбора типа действия следует кликнуть по одному из пунктов. В результате откроется окно редактирования действия:

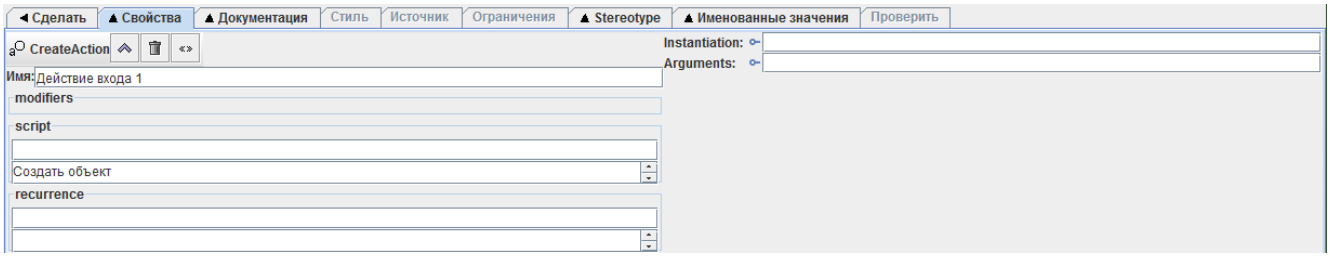


Рис.4. Свойства действия входа (тип: создание объекта).

В этом окне можно задать имя действия и, собственно, содержание действия (группа script). Именно содержание многострочного поля script (в примере – «Создать объект») отобразится на диаграмме. В однострочном поле этой группы (первом по порядку) можно указать язык скрипта.

В качестве действия можно выбрать последовательность. Редактирование последовательности действий выглядит следующим образом:

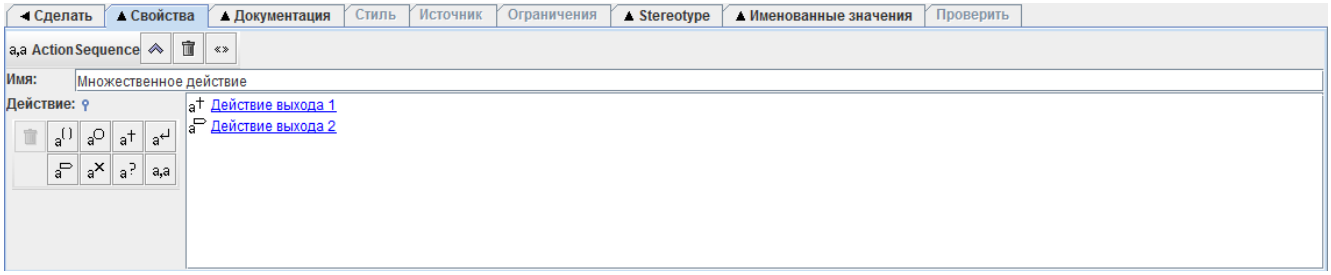


Рис.5. Свойства деятельности выполнения (тип: последовательность действий)

С помощью меню слева, аналогичного меню создания действий, можно добавлять действия к последовательности. Для редактирования отдельных действий нужно кликнуть по произвольному действию 2 раза. Редактирование каждого отдельного действия аналогично редактированию единичных действий.

Псевдосостояния

Псевдосостояния – начальное и конечное состояния объекта или процесса. Каждая диаграмма состояний должна начинаться начальным состоянием и заканчиваться конечным. Для добавления на диаграмму начального и конечного состояний используются кнопки и соответственно.

Рассмотрим следующий пример. 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.

Разместим все состояния объекта на диаграмме, в том числе начальное и конечное.

Определим некоторые действия, которые выполняются в данных состояниях.

- на входе в состояние Инициализация – действие «Сохранить дату заказа»;
- деятельность состояния Инициализация – действие «Внести информацию о клиенте»;
- на выходе из состояния Открыт – действие создания пункта заказа;
- на выходе из состояния Отменен – действие «Сохранить дату отмены»;
- деятельность состояния Закрыт – действие «Выписать счет».

Действие добавления пунктов заказа, а также удаления отмененного заказа, пока опустим. С помощью инструмента Broom можно выровнять элементы относительно друг друга.

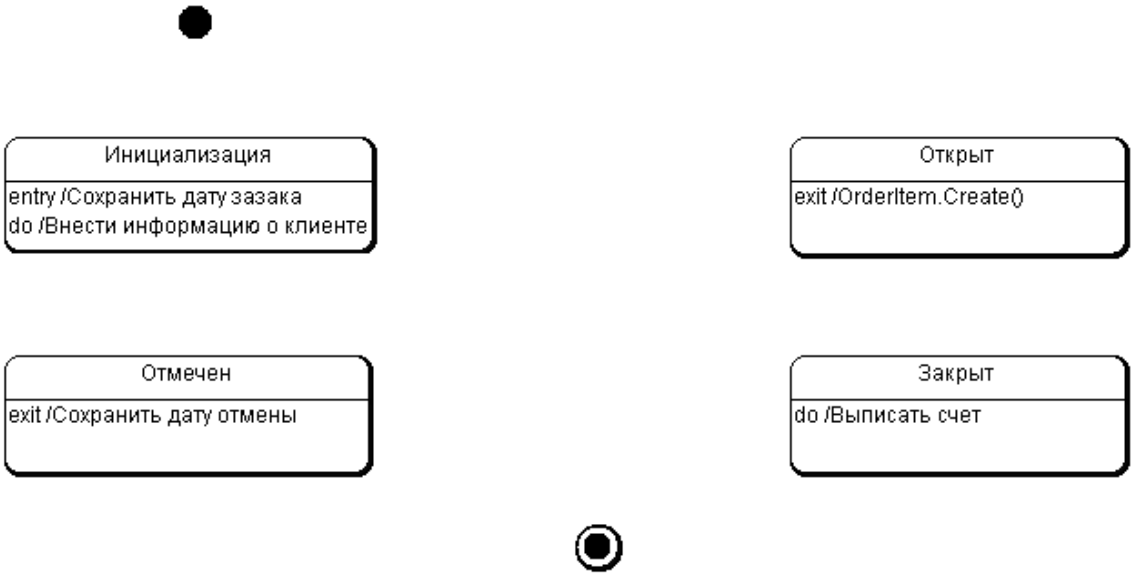


Рис.6. Состояния объекта Entry.

Добавление переходов

Переходы добавляются на диаграмму с помощью инструмента New Transition. После добавления перехода или при его выделении, под областью диаграммы открывается окно редактирования перехода.

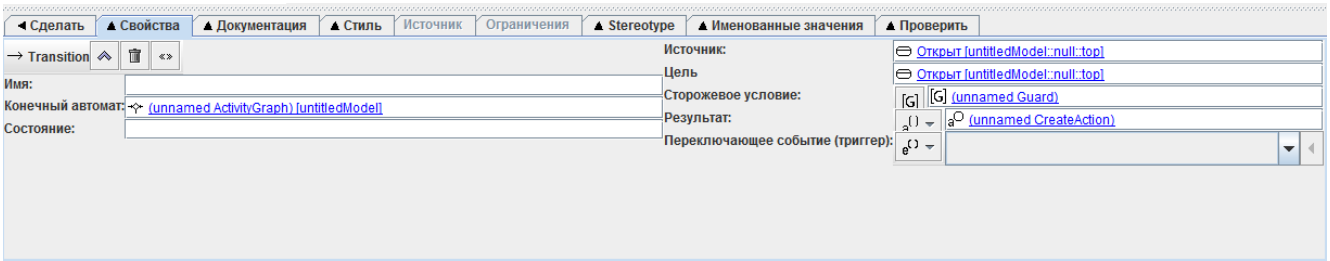
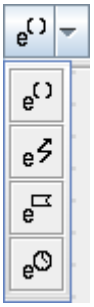


Рис.7. Свойства перехода.

Наиболее важными свойствами связей являются Переключающее событие (триггер), Результат (действие) и Сторожевое условие (выражение).

Триггер – некоторое событие, которое возбуждает изменение состояния объекта. Для того чтобы задать переключающее событие, нужно раскрыть выпадающее меню и выбрать его тип. Всего в ArgoUML определены 4 типа событий:



- событие вызова;

- событие изменения;

- событие сигнала;

- событие времени.

Для события можно задать имя и выражение, которым оно характеризуется. Окно редактирования события выглядит следующим образом:

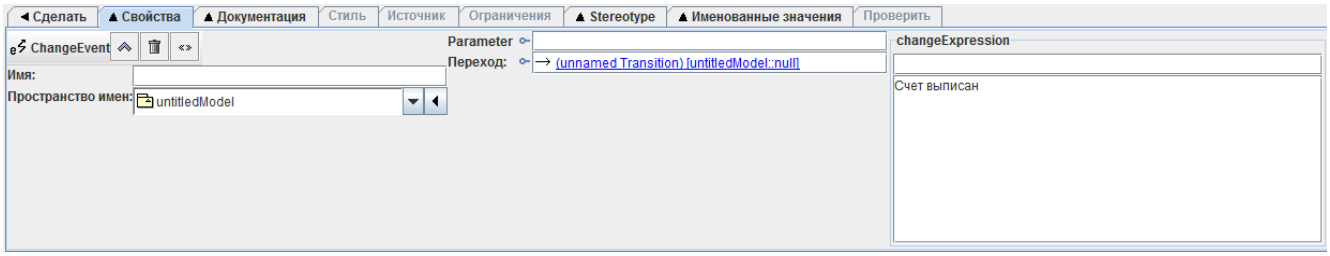


Рис.7. Свойства события «Счет выписан» (тип: событие изменения).

Сторожевое условие (*Guard*) – некоторое выражение, которое определяет семантику срабатывания перехода. Если условие истинно – переход выполняется, если же оно ложно – переход невозможен, и при отсутствии других переходов объект не может перейти в целевое состояние.

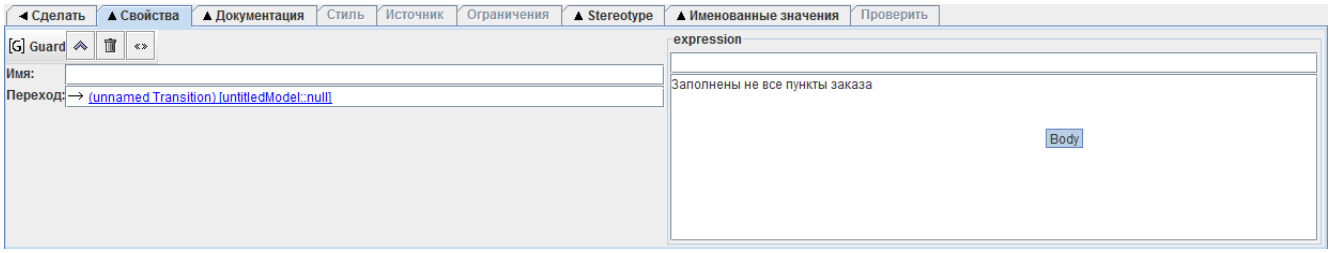


Рис.8. Сторожевое условие

Наконец, для перехода можно задать результат. Результат перехода – действие, полностью аналогичное действиям входа/выхода или деятельности состояний. Определение действий в переходах характерно для автоматов Мили. Процесс задания действий при переходах аналогичен таковому для состояний.

Вернемся к примеру. Добавим переходы между состояниями. Необходимо создать следующие триггеры:

- Заказ создан (событие сигнала) – переход из начального состояния в состояние «Инициализация»;

- Заказ отменен (событие изменения) – переход из состояния «Открыт» в состояние «Отменен»;

- Счет выписан (событие изменения) – переход из состояния «Закрыт» в конечное состояние.

Также определим несколько сторожевых условий:

- Инициализация завершена – условие перехода из состояния «Инициализация» в состояние «Открыт»;

- Заполнены не все пункты заказа – условие перехода из состояния «Открыт» в себя же;

- Заполнены все позиции заказа – условие перехода из состояния «Открыт» в состояние «Закрыт».

Необходимо задать недостающие действия для переходов:

- Добавление пункта заказа – действие создания при переходе из состояния «Открыт» в себя;

- Удаление записи (OrderItem.Delete()) – действие уничтожения при переходе из состояния «Отменен» в конечное состояние.

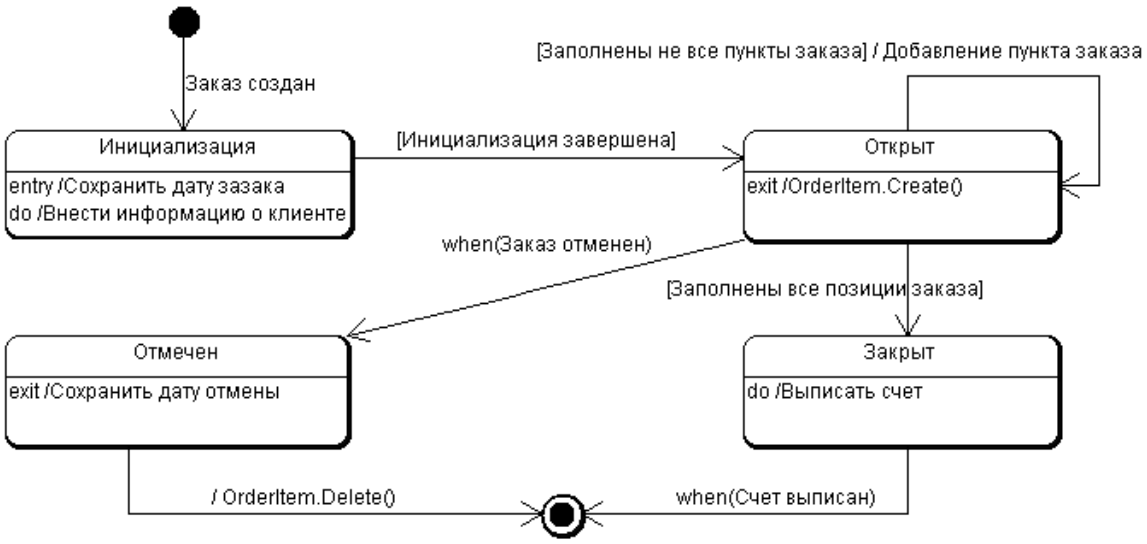


Рис.9. Диаграмма состояний объекта Entry.

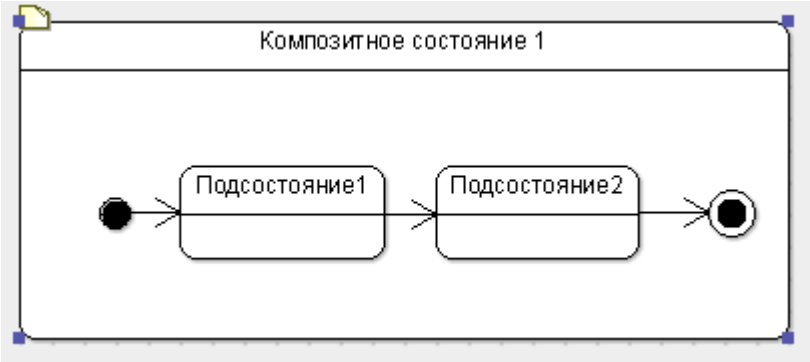
Отложенные события

В свойствах состояния с помощью элемента *Deferrable Events* могут задаваться отложенные события. Такие события не обрабатываются внутри состояния, вместо этого они записываются в очередь. На входе в следующее состояние все отложенные события, которые находятся на данный момент в очереди, обрабатываются, при этом порядок обработки не специфицируется. Отложенные события могут инициализировать переходы. Отложенные события следует использовать осторожно. Визуально отложенные события на диаграммах ArgoUML не отображаются.

Композитные состояния.

С помощью кнопки *Композитное состояние* на диаграмму можно добавлять одноименные элементы (также называемые *составными состояниями*). Композитное состояние обладает всеми свойствами обычных состояний. Помимо этого, оно может содержать другие состояния, называемые *подсостояниями*. Для добавления подсостояний можно перетаскивать их внутрь композитного состояния мышью или сразу создавать их в контейнере, кликая внутри области композитного состояния.

Вот так выглядит композитное состояние из двух подсостояний + псевдосостояний:



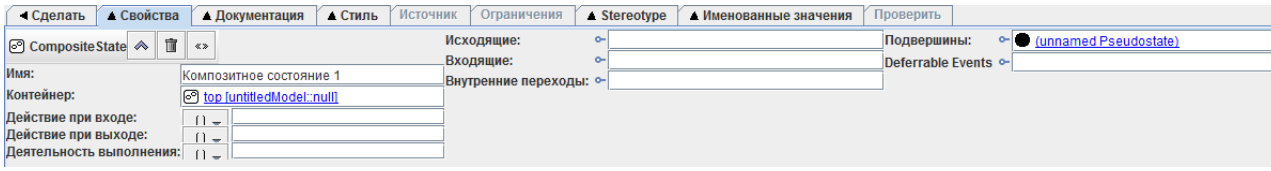


Рис.10. Пример композитного (составного) состояния.

Навигация по подсостояниям может осуществляться через элемент «Подвершины» окна свойств композитного состояния.

Композитные состояния часто применяют для описания подавтоматов. Однако композитное состояние может содержать как один, так и несколько параллельных подавтоматов. Обычно при описании параллельных последовательностей подсостояний композитное состояние разделяют на регионы пунктирной линией, но это не является обязательным и в ArgoUML не реализовано. Подсостояния могут быть вовсе не определены, если необходимо скрыть внутреннюю структуру композитного состояния (В ArgoUML композитное состояние, в котором не определены подсостояния, внешне не отличается от обычного состояния).

Исторические состояния

В силу разных причин выполнение композитного состояния может прерываться. Историческое состояние (history state) применяется в контексте композитного состояния. Оно используется для запоминания того из последовательных подсостояний, которое было текущим в момент выхода из композитного состояния. Существует две разновидности исторического состояния: недавнее (shallow) и давнее (deep).

Shallow history всегда является первым подсостоянием в композитном состоянии (переход извне всегда осуществляется в это состояние, оно же заменяет начальное псевдосостояние). При первом переходе в композитное состояние, историческое состояние всегда пусто, и выполняет роль начального (при этом начальное псевдосостояние все равно отображается на диаграмме). При всех последующих переходах, историческое состояние автоматически осуществляет переход в то подсостояние, которое было текущим в момент выхода из этого композитного состояния. При достижении конечного псевдосостояния композитным состоянием, историческое состояние обнуляется и снова будет выполнять роль начального псевдосостояния. Недавнее историческое состояние может запоминать только текущее подсостояние того автомата, к которому относится. При этом запомненное подсостояние, в свою очередь, может также быть композитным.

Deep history применяется для запоминания всех подсостояний любого уровня вложенности. Семантика аналогична недавнему историческому состоянию.

Следующая диаграмма показывает семантику исторического состояния. В данном примере композитное состояние «Обработка» может прерываться в подсостояниях «Проверка 1» и «Проверка 2». После возврата в состояние «Обработка» управление автоматически перейдет в то состояние, на котором обработка была прервана.

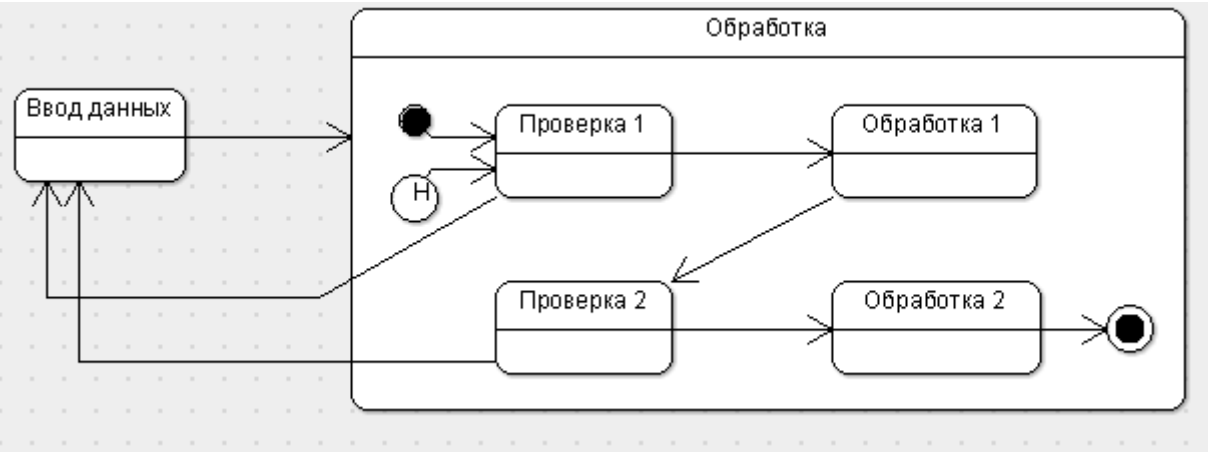


Рис.11. Пример использования исторического состояния.

Контрольные вопросы

1. Приведите определения следующих понятий: автомат, состояние, событие, действие, переход.
2. Для чего предназначена диаграмма состояний?
3. К каким элементам системы может быть привязана диаграмма состояний?
4. Как изображается диаграмма состояний?
5. Какие существуют типы состояний?
6. Для чего служат действия, события, сторожевые условия?
7. Какие типы исторических состояний бывают? В чем их различие?

13. Диаграммы прецедентов.

Цель работы: получить навыки построения диаграмм прецедентов.

Задание:

1. создать главную диаграмму прецедентов, задав на ней варианты использования и актеров;
2. добавить отношения между актерами и вариантами использования;
3. создать дополнительную диаграмму прецедентов;
4. добавить описания к актерам и вариантам использования;
5. для каждого варианта использования задать поток событий в виде отдельного файла и прикрепить его к варианту использования.

Содержание отчета:

1. созданные диаграммы прецедентов;
2. краткое описание каждого актера и прецедента;
3. описание потока событий для каждого варианта использования.

Теоретическая часть

Визуальное моделирование с использованием нотации UML можно представить как некоторый процесс поуровнего спуска от наиболее общей и абстрактной концепту-

альной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы. Для достижения этих целей вначале строится модель в форме **диаграммы вариантов использования** (**UseCasediagram)**, которая описывает функциональное назначение системы. Диаграмма UseCase является исходным концептуальным представлением (концептуальной моделью) системы в процессе ее проектирования и разработки. Она определяет поведение системы с точки зрения пользователя и рассматривается как главное средство для первичного моделирования динамики системы. Диаграмма UseCase используется для выяснения требований к разрабатываемой системе, фиксации этих требований в форме, которая позволит вести дальнейшую разработку. Диаграммы UseCase также называют диаграммами прецедентов.

Разработка диаграммы вариантов использования преследует следующие цели:

- определение общих границ и контекста моделируемой предметной области на начальных этапах проектирования системы;
- формулировка общих требований к функциональному поведению проектируемой системы;
- разработка исходной концептуальной модели системы для ее последующей детализации в форме логических и физических моделей;
- подготовка исходной документации для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Основное назначение диаграмм UseCase – определение требований заказчика к будущему программному приложению. Суть данной диаграммы состоит в следующем: проектируемая система представляется в форме так называемых вариантов использования, с которыми взаимодействуют некоторые внешние сущности или актеры. При этом актером или действующим лицом называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит разработчик. В свою очередь вариант использования служит для описания сервисов, которые система предоставляет актеру, т.е. каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. Каким образом будет реализовано взаимодействие актеров с системой и собственно вариантов использования, данная диаграмма не показывает. В этом смысле диаграмма UseCase ассоциируется с моделью "черного ящика".

Диаграмма UseCase представляет собой граф специального вида. В состав диаграмм входят элементы UseCase (варианты использования), актеры, а также отношения (зависимости, обобщения, ассоциации и др.). Актеры и элементы UseCase являются вершинами графа. Актеры представляют внешний мир, нуждающийся в работе системы, элементы UseCase представляют действия, выполняемые системой в интересах актеров. Отношения – дуги графа. Диаграммы UseCase могут включать примечания и ограничения и содержать пакеты, используемые для группировки элементов модели в крупные фрагменты.

Актер – это роль объекта вне системы, который прямо взаимодействует с ее частью – конкретным элементом (элементом UseCase). Различают актеров и пользователей. Пользователь – это физический объект, который использует систему. Он может играть несколько ролей и поэтому может моделироваться несколькими актерами. Справедливо и другое – актером могут быть разные пользователи. Например, для коммерческого летательного аппарата можно выделить двух актеров: пилота и кассира. Сидоров – пользователь, который иногда действует как пилот, а иногда как кассир. В зависимости от роли Сидоров взаимодействует с разными элементами UseCase.

Элемент UseCase – это описание последовательности действий (или нескольких последовательностей), которые выполняются системой и производят для отдельного актера видимый результат. Один актер может использовать несколько элементов UseCase, и наоборот один элемент UseCase может иметь несколько актеров, использующих его. Каж-

дый элемент UseCase задает определенный путь использования системы. Набор всех элементов UseCase определяет полные функциональные возможности системы.

Между актером и элементом UseCase возможен только один вид ***отношения*** – ассоциация, отображающая их взаимодействие. Как и любая другая ассоциация, она может быть помечена именем ролями и мощностью. Между актерами допустимо отношение обобщения, означающее, что экземпляр потомка может взаимодействовать с такими же разновидностями экземпляров UseCase, что и экземпляр родителя. Между элементами UseCase определены отношение обобщения и две разновидности отношения зависимости – включения и расширения. Отношение обобщения фиксирует, что потомок наследует поведение родителя. Кроме того, потомок может дополнить или переопределить поведение родителя. Элемент UseCase, являющийся потомком, может замещать элемент UseCase, являющийся родителем, в любом месте диаграммы. Отношение включения между элементами UseCase означает, что базовый элемент UseCase явно включает поведение другого элемента UseCase в точке, которая определена в базе. Включаемый элемент UseCase никогда не используется самостоятельно – его конкретизация может быть только частью другого, большего элемента UseCase. Отношение включения является примером отношения делегации. При этом в отдельное место (включаемый элемент UseCase) помещается определенный набор обязанностей системы. Далее остальные части системы могут агрегировать в себя эти обязанности (при необходимости). Отношение расширения между элементами UseCase означает, что базовый элемент UseCase неявно включает поведение другого элемента UseCase в точке, которая определяется косвенно расширяющим элементом UseCase.

Элемент UseCase описывает, что должна делать система, но не определяет, как она должна это делать. При моделировании это позволяет отделять внешнее представление системы от ее внутреннего представления. Поведение элемента UseCase описывается потоком событий. Начальное описание выполняется в текстовой форме, прозрачной для пользователя системы. В потоке событий выделяют: основной поток и альтернативные потоки поведения; как и когда стартует и заканчивается элемент UseCase; когда элемент UseCase взаимодействует с актерами; какими данными обмениваются актер и система. Для уточнения и формализации потоков используют **диаграммы последовательности.** С помощью диаграммы последовательностей можно описать полный контекст взаимодействий как своеобразный график "жизни" всей совокупности объектов, взаимодействующих между собой для реализации варианта использования программной системы, достижения бизнес-цели или выполнения какой-либо задачи. Обычно одна диаграмма последовательности определяет главный поток в элементе UseCase, а другие диаграммы – потоки исключений. В общем случае каждый элемент UseCase описывает набор последовательностей, в котором каждая последовательность представляет возможный поток событий. Каждая последовательность называется сценарием. Сценарий – конкретная последовательность действий, которая иллюстрирует поведение. Сценарии являются для элемента UseCase почти тем же, чем являются экземпляры для класса. Говорят, что сценарий – это экземпляр элемента UseCase.

1. Предварительные действия - создание новой модели

Примечание: В этом и последующих примерах мы будем проектировать систему для предметной области "*Предприятие по сборке и продаже компьютеров*". Работа с ArgoUml начинается с создания модели. При запуске программы автоматически создается поле для создания Диаграммы классов (рис. 1) .

Для создания поля для диаграммы прецедентов выбираем соответствующий значок, и появляется набор инструментов для этой диаграммы (рис.2).

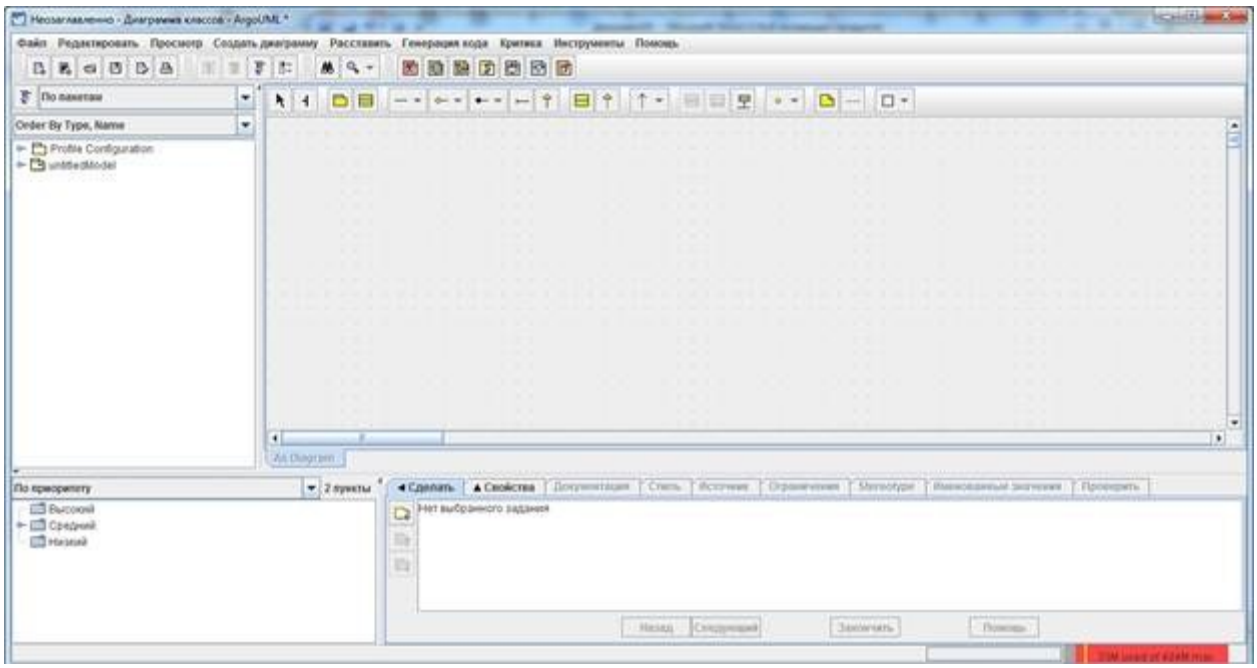


Рис. 1. Окно с автоматически созданной формой для диаграммы классов.

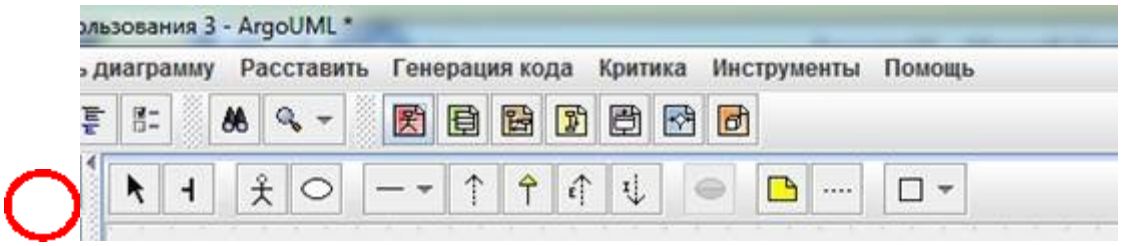


Рис. 2.

2. Создание главной диаграммы прецедентов

Над рабочей областью существует панель с инструментами. Для создания диаграммы надо нажать на требуемый элемент и нажать в рабочей области на место, где он будет размещаться. В этом месте будет создан элемент выбранного типа (рис. 3).

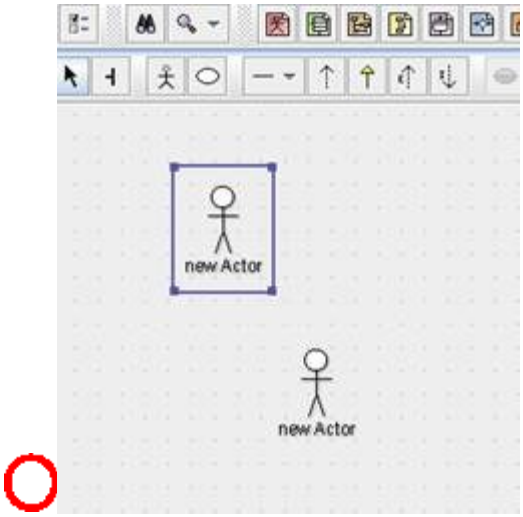


Рис. 3.

Для нашей предметной области мы выделили следующих актеров:

Актер	**Краткое описание**
Менеджер по работе с клиентами	Сотрудник, который общается с заказчиком и работает с заказом
Менеджер по снабжению	Сотрудник, который занимается закупкой необходимых комплектующих
Инженер по сборке настольных компьютеров	Сотрудник, который занимается сборкой настольных компьютеров
Инженер по сборке ноутбуков	Сотрудник, который занимается сборкой ноутбуков
Инженер по тестированию	Сотрудник, который занимается тестированием собранных компьютеров
Завскладом	Сотрудник, который заведует складом комплектующих

Рассмотрим теперь, какие возможности должна предоставлять наша система:

- актер *Менеджер по работе с клиентами* использует систему для оформления, редактирования заказов и управления информацией о клиентах предприятия;
- актер *Менеджер по снабжению* использует систему для просмотра перечня необходимых для закупки комплектующих и ведения информации о снабжении;
- актер *Инженер по сборке настольных компьютеров* использует систему для просмотра нарядов на сборку персональных компьютеров, для заказа комплектующих со склада и отметки о ходе выполнения работы;
- актер *Инженер по сборке ноутбуков* использует систему для просмотра нарядов на сборку ноутбуков, для заказа комплектующих со склада и отметки о ходе выполнения работы;
- актер *Инженер по тестированию* использует систему для просмотра нарядов на тестирование собранной продукции и отметки о ходе выполнения работы;
- актер *Завскладом* использует систему для учета поступления и выдачи комплектующих.

На основании вышеизложенного можно выделить следующие прецеденты:

Прецедент	**Краткое описание**
Работа с заказом	Запускается менеджером по работе с клиентами. Позволяет вносить, изменять, удалять или просматривать заказ.
Управление информацией о клиенте	Запускается менеджером по работе с клиентами. Позволяет добавлять, изменять или удалять клиентов, а также просматривать информацию о клиентах.
Управление информацией о поставщиках	Запускается менеджером по снабжению. Позволяет добавлять, изменять или удалять поставщиков, а также просматривать информацию о поставщиках.
Управление информацией о комплектующих	Запускается менеджером по снабжению. Позволяет просматривать информацию о комплектующих, производить анализ их расходования, прогнозировать необходимое их количество и делать заказ.
Сборка компьютеров	Запускается инженером по сборке. Позволяет просматривать наряды на сборку компьютеров и делать отметки о ходе выполнения работы.
Требование необходимых комплектующих	Запускается инженером по сборке. Предназначено для затребования необходимых комплектующие со склада.
Тестирование компью-	Запускается инженером по тестированию. Позволяет просмотреть

Прецедент	Краткое описание
теров	список компьютеров, подлежащих тестированию и сделать отметки о ходе выполнения работ.
Учет поступления и выдачи комплектующих	Запускается завскладом. Позволяет вести учет поступления и выдачи запчастей и комплектующих.

Созданная главная диаграмма прецедентов показана на рис. 4:

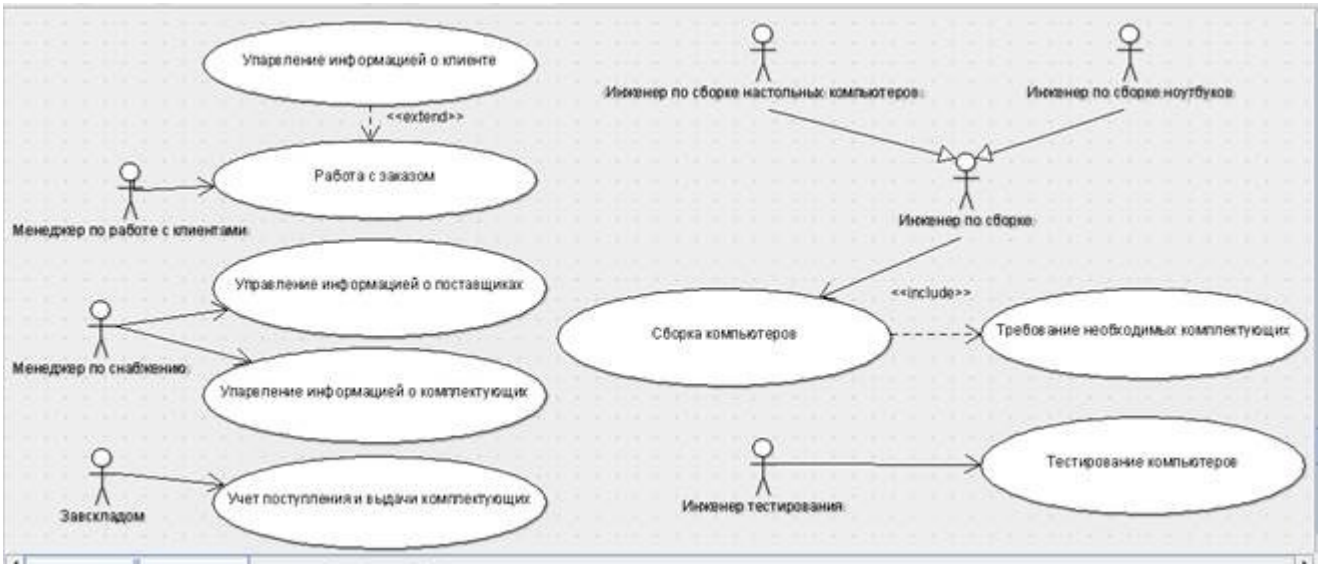


Рис. 4. Главная диаграмма прецедентов

Рассмотрим теперь отношения между актерами и прецедентами. В языке UML возможен только один тип отношений между актером и прецедентом - отношение коммуникации. Поэтому всех актеров мы связали с прецедентами отношением *Unidirectional Association*. Поскольку другой тип отношений здесь мы задать не может, то стереотип *communicate* можно не указывать (он неявно подразумевается).

Для прецедента *Сборка компьютеров* не имеет значение какой именно актер будет с ним взаимодействовать - *Инженер по сборке настольных компьютеров* или *Инженер по сборке ноутбуков.* Поэтому мы ввели еще одного актера - *Инженер по сборке*, с которым связали первых двух актеров отношением обобщения (*Generalization*).

Отношение между прецедентами *Работа с заказом* и *Управление информацией о клиенте* - отношение расширения, поскольку когда актер *Менеджер по работе с клиентами* работает с заказом (оформляет, меняет и т.д.), то не всегда при этом он управляет информацией о клиентах.

Отношение между прецедентами *Сборка компьютеров* и *Требование необходимых комплектующих* - отношение включения, поскольку для сборки компьютеров обязательно нужно заказывать необходимые комплектующие со склада.

3. Поток событий для прецедентов главной диаграммы прецедентов

Потоки событий для прецедентов будем описывать по следующему шаблону:

- X.1 предусловия;
- X.2 главный поток;
- X.3 под-потоки;
- X.4 альтернативные потоки;
- X.5 постусловия.

где X - число от единицы до количества прецедентов.

Поток событий для прецедента «Работа с заказом».

1.1 Предусловия.

Если заказ оформляется для нового клиента, то под-поток *добавить нового клиента (Add a New Client)* прецедента *Управление информацией о клиенте*должен быть выполнен перед его началом.

1.2 Главный поток.

Прецедент начинает выполняться, когда менеджер подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *добавить (Add)*, *изменить (Change), удалить (Delete)*, *просмотреть (View)* или *выйти (Exit).*

Если выбрана операция *добавить (Add)*, S-1: выполняется поток *добавить новый заказ (Add a New Order)*.

Если выбрана операция *изменить (Change)*, S-2: выполняется поток *изменить заказ (Change Order)*.

Если выбрана операция *удалить (Delete)*, S-3: выполняется поток *удалить заказ (Delete Order).*

Если выбрана операция *просмотреть (View)*, S-4: выполняется поток *просмотреть заказ (View Order).*

Если выбрана операция *выйти (Exit)* прецедент завершается.

1.3 Под-потоки.

S-1: *добавить новый заказ (Add a New Order)*

Система отображает диалоговое окно, содержащее поле, в котором менеджер должен выбрать тип компьютера (настольный или ноутбук). Пользователь выбирает необходимый тип. Система отображает поле для выбора клиента и список возможных комплектующих для выбранного типа компьютера, в котором менеджер отмечает выбранные клиентом комплектующие. Менеджер заполняет поля (E-2). Система запоминает введенные данные и распечатывает счет для оплаты. Затем прецедент начинается сначала.

S-2: *изменить заказ (Change Order)*

Система отображает диалоговое окно, содержащее список заказов и поле для ввода номера заказа. Менеджер выбирает необходимый заказ из списка или вводит номер заказа в поле (E-3). Система отображает информацию о данном заказе. Менеджер делает необходимые изменения (E-2). Система запоминает введенные данные. Затем прецедент начинается сначала.

S-3: *удалить заказ (Delete Order)*

Система отображает диалоговое окно, содержащее список заказов и поле для ввода номера заказа. Менеджер выбирает необходимый заказ из списка или вводит номер заказа в поле (E-3). Система удаляет выбранный заказ (E-4). Затем прецедент начинается сначала.

S-4: *просмотреть заказ (View Order)*

Система отображает диалоговое окно, содержащее список заказов и поле для ввода номера заказа. Менеджер выбирает необходимый заказ из списка или вводит номер заказа в поле (E-3). Система отображает информацию о выбранном заказе. Когда менеджер просмотрит информацию, прецедент начнется сначала.

1.3 Альтернативные потоки

E-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.

E-2: выбраны не все комплектующие, необходимые для сборки компьютера или комплектующих нет в наличии. Менеджер должен изменить состав компьютера или завершить прецедент.

E-3: введен неправильный номер заказа. Менеджер должен повторить ввод или завершить прецедент.

E-4: система не может удалить заказ. Информация сохраняется, система удалит заказ позже. Выполнение прецедента продолжается.

Поток событий для прецедента «Управление информацией о клиенте».

2.1 Предусловия.

Если заказ оформляется для нового клиента, то под-поток *добавить нового клиента (Add a New Client)* прецедента *Управление информацией о клиенте*должен быть выполнен перед его началом.

2.2 Главный поток.

Прецедент начинает выполняться, когда менеджер подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *добавить (Add)*, *изменить (Change), удалить (Delete)*, *просмотреть (View)* или *выйти (Exit).*

Если выбрана операция *добавить (Add)*, S-1: выполняется поток *добавить нового клиента (Add a New Client)*.

Если выбрана операция *изменить (Change)*, S-2: выполняется поток *изменить данные о клиенте (Change Client Data)*.

Если выбрана операция *удалить (Delete)*, S-3: выполняется поток *удалить клиента (Delete Client).*

Если выбрана операция *просмотреть (View)*, S-4: выполняется поток *просмотреть данные о клиенте (View Client Data).*

Если выбрана операция *выйти (Exit)* прецедент завершается.

2.2 Под-потоки.

S-1: *добавить нового клиента (Add a New Client)*Система отображает диалоговое окно, содержащее поля для ввода данных о новом клиенте. Пользователь заполняет поля (E-2). Система запоминает введенные данные. Затем прецедент начинается сначала.

S-2: *изменить данные о клиенте (Change Client Data)*

Система отображает диалоговое окно, содержащее список клиентов и поле для ввода номера клиента. Менеджер выбирает необходимого клиента из списка или вводит его номер в поле (E-3). Система отображает информацию о данном клиенте. Менеджер делает необходимые изменения (E-2). Система запоминает введенные данные. Затем прецедент начинается сначала.

S-3: *удалить клиента (Delete Client)*

Система отображает диалоговое окно, содержащее список клиентов и поле для ввода номера клиента. Менеджер выбирает необходимого клиента из списка или вводит его номер в поле (E-2). Система удаляет выбранного клиента (E-4). Затем прецедент начинается сначала.

S-4: *просмотреть данные о клиенте (View Client Data)*

Система отображает диалоговое окно, содержащее список клиентов и поле для ввода номера клиента. Менеджер выбирает необходимого клиента из списка или вводит его номер в поле (E-3). Система отображает информацию о выбранном клиенте. Когда менеджер просмотрит информацию, прецедент начнется сначала.

2.3 Альтернативные потоки

E-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.

E-2: заполнены не все поля. Менеджер должен заполнить незаполненные поля или завершить прецедент.

E-3: введен неправильный номер клиента. Менеджер должен повторить ввод или завершить прецедент.

E-4: система не может удалить клиента. Информация сохраняется, система удалит клиента позже. Выполнение прецедента продолжается.

Поток событий для прецедента «Учет поступления и выдачи комплектующих.

3.1 Предусловия.

3.2 Главный поток.

Прецедент начинает выполняться, когда завскладом подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *добавить (Add)*, *отметить (Mark)* или *выйти (Exit).*

Если выбрана операция *добавить (Add)*, S-1: выполняется поток *внести поступившие комплектующие (Add a New Components)*.

Если выбрана операция *отметить (Mark)*, S-2: выполняется поток *сделать отметку о выдаче комплектующих (Mark Components)*.Если выбрана операция *выйти (Exit)* прецедент завершается.

3.3 Под-потоки.

S-1: *внести поступившие комплектующие (Add a New Components)*

Система отображает диалоговое окно, содержащее поля для ввода наименования комплектующих, их количества, поставщика. Завскладом заполняет указанные поля (E-2). Система запоминает введенные данные. Затем прецедент начинается сначала.

S-2: *сделать отметку о выдаче комплектующих (Change Order)*

Система отображает список комплектующих, находящихся на складе. Завскладом напротив нужных комплектующих вводит количество выданных (E-3). Система запоминает введенные данные. Затем прецедент начинается сначала.

3.4 Альтернативные потоки

E-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.

E-2: заполнены не все поля. Пользователь должен заполнить пропущенные поля или завершить прецедент.

E-3: указано количество выданных комплектующих, превышающее их количество на складе. Пользователь должен повторить ввод или завершить прецедент.

Поток событий для прецедента «Сборка компьютеров».

4.1 Предусловия.

4.2 Главный поток.

Прецедент начинает выполняться, когда инженер по сборке подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *просмотреть (View)*, *отметить (Mark)* или *выйти (Exit).*

Если выбрана операция *просмотреть (View)*, S-1: выполняется поток *Просмотреть наряд на сборку компьютера (View an Make Computer Order)*.

Если выбрана операция *отметить (Mark)*, S-2: выполняется поток *сделать отметку о статусе собираемого компьютера по наряду (Mark Computer)*.Если выбрана операция *выйти (Exit)* прецедент завершается.

4.3 Под-потоки.

S-1: *Просмотреть наряд на сборку компьютера (View an Make Computer Order)*Система отображает диалоговое окно, содержащее список нарядов и поле для ввода номера наряда. Инженер выбирает необходимый наряд из списка или вводит его номер в поле (E-2). Система отображает информацию о выбранном наряде. Когда инженер просмотрит информацию, прецедент начнется сначала.

S-2: *сделать отметку о статусе собираемого компьютера (Mark Computer)*Система отображает диалоговое окно, содержащее список нарядов. Возле необходимого наряда инженер делает отметку о статусе компьютера по данному наряду. Инженер сохраняет изменения. Затем прецедент начинается сначала.

4.4 Альтернативные потоки

E-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.E-2: заполнены не все поля. Пользователь должен заполнить пропущенные поля или завершить прецедент.E-3: введен неправильный номер наряда. Инженер должен повторить ввод или завершить прецедент.

Поток событий для прецедента «Требование необходимых комплектующих».

5.1 Предусловия.

5.2 Главный поток.

Прецедент начинает выполняться, когда инженер по сборке подключается к системе и вводит свое имя и пароль. Система проверяет правильность пароля (E-1) и выводит возможные варианты действий: *просмотреть (View)*, *затребовать (Order)* или *выйти (Exit)*.

Если выбрана операция *просмотреть (View)*, S-1: выполняется поток *просмотреть затребованные комплектующие на складе (View Ordered Components on Warehouse)*.

Если выбрана операция *затребовать (Order)*, S-2: выполняется поток *затребовать необходимые комплектующие на складе (Order Required Components on Warehouse)*.

Если выбрана операция *выйти (Exit)* прецедент завершается.

5.3 Под-потоки.

S-1: *Просмотреть затребованные комплектующие на складе (View Ordered Components on Warehouse)*

Система отображает следующую информацию обо всех сделанных заказах данным инженером по сборке: дата затребования, наименование комплектующих, их количество, заказ выполнен или нет. Когда инженер по сборке просмотрел список, он уведомляет систему. Прецедент начинается сначала.

S-2: *затребовать необходимые комплектующие на складе (Order Required Components on Warehouse)*

Система отображает диалоговое окно, содержащее поля для ввода списка необходимых комплектующих и их количества. Инженер по сборке заполняет его. Система запоминает введенные данные. Затем прецедент начинается сначала.

5.4 Альтернативные потоки

Е-1: введено неправильное имя или пароль. Пользователь должен повторить ввод или завершить прецедент.

Описание потоков событий для прецедентов *Управление информацией о поставщиках* и *Управление информацией о комплектующих* аналогичноописанию для прецедента *Управление информацией о клиенте;* для прецедента *Тестирование компьютеров* - прецеденту *Сборка компьютеров.*

3. Создание дополнительной диаграммы прецедентов.

Как видно из описания потока событий для всех прецедентов каждый из них включает проверку пользователя. Проверка осуществляется единообразно для любого прецедента. Поэтому ее можно представить в виде отдельного прецедента *Аутентификация пользователя*, связанного отношением включения со всеми остальными. Результат создания диаграммы показан на рис. 5:

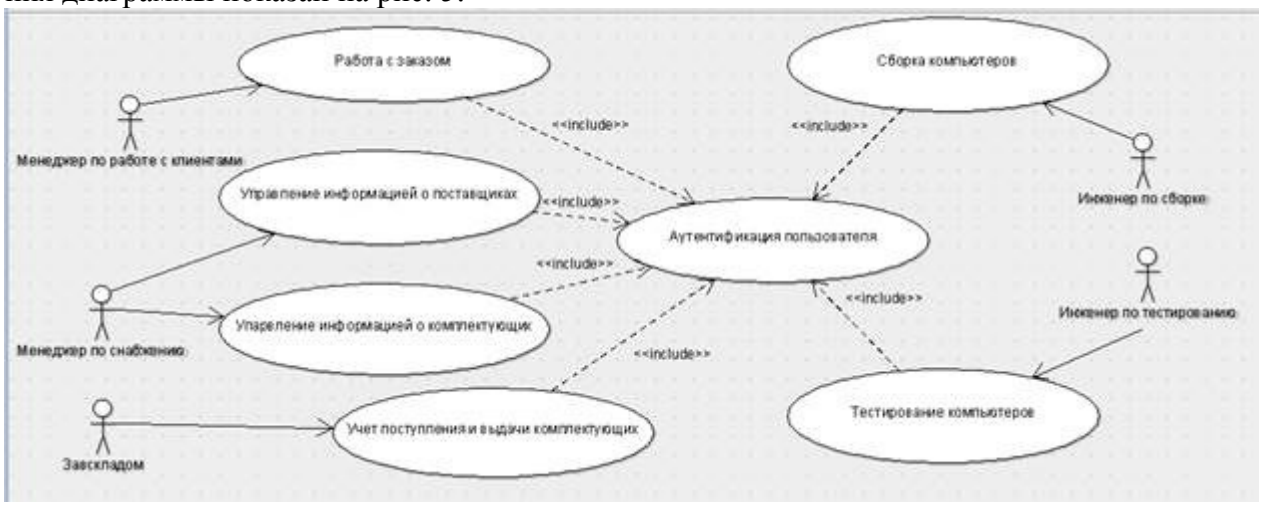


Рис. 5. Дополнительная диаграмма прецедентов

Методика количественной оценки и сравнения диаграмм UML строится на присвоении элементам диаграмм оценок, зависящих от их информационной ценности, а также от вносимой ими в диаграмму дополнительной сложности. Ценность отдельных элементов меняется в зависимости от типа диаграммы, на которой они находятся.

Количественную оценку диаграммы можно провести по следующей формуле:

$$S = \frac{\sum S_{Obj} + \sum S_{Lnk}}{1 + Obj + \sqrt{T_{Obj} + T_{Lnk}}}$$

S - оценка диаграммы

S_{Obj} - оценки для элементов диаграммы

S_{Lnk} - оценки для связей на диаграмме

Obj - число объектов на диаграмме

T_{Obj} - число типов объектов на диаграмме

T_{Lnk} - число типов связей на диаграмме

Если диаграмма содержит большое количество связей одного типа, то число и тип связей можно не учитывать и формула расчета приводится к виду:

$$S = \frac{\sum S_{Obj}}{1 + Obj + \sqrt{T_{Obj}}}$$

Если на диаграмме показаны атрибуты и операции классов, можно учесть их при расчете, при этом оценка прибавляется к оценке соответствующего класса:

$$S_{cls} = \frac{\sqrt{Op} + \sqrt{Art}}{0{,}3 * (Op + Art)}$$

S_{cls} - оценка операций и атрибутов для класса

Op - число операций класса

Art - число атрибутов класса

Приведем оценки для различных типов элементов и связей:

Основные элементы языка UML:

Тип элемента	**Оценка для элементов**
Класс	
Интерфейс	
Прецедент	
Компонент	
Узел	
Процессор	
Взаимодействие	
Пакет	
Состояние	
Примечание	

Основные типы связей языка UML:

Тип связи	**Оценка для связи**
Зависимость	
Ассоциации	
Агрегирование	
Композиция	
Обобщение	
Реализация	

Остальные типы связей должны рассматриваться как ассоциации.

Недостатком диаграммы является как слишком низкая оценка (при этом диаграмма недостаточно информативна), так и слишком высокая оценка (при этом диаграмма обычно слишком сложна для понимания).

Диапазоны оценок для диаграмм UML:

Тип диаграммы	**Диапазон оценок**
Классов – с атрибутами и операциями	5 – 5,5
Классов – без атрибутов и операций	3 – 3,5
Компонентов	3,5 – 4
Вариантов использования	2,5 – 3
Развертывания	2 – 2,5
Последовательности	3 – 3,5
Кооперативная	3,5 – 4
Пакетов	3,5 – 4

Состояний 2,5 – 3

Пример оценки простой диаграммы классов по данной методике:

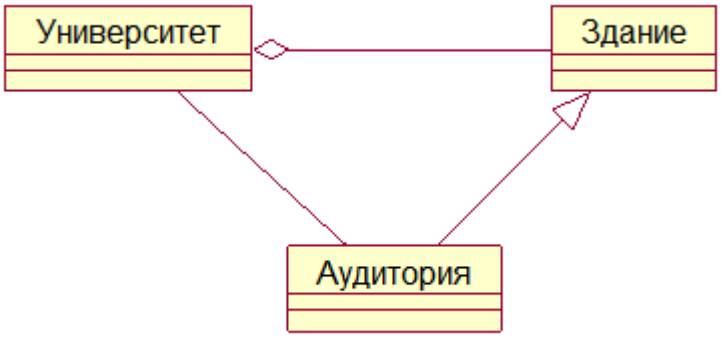


Диаграмма содержит три класса без операций и атрибутов; следовательно, T_{Obj}=1,

$\sum S_{Obj}$=15 и Obj=3. В качестве связей используется ассоциация, агрегирование и обобщение; следовательно, $\sum S_{Lnk}$=6 и T_{Lnk}=3.

$$S = \frac{\sum S_{Obj} + \sum S_{Lnk}}{1 + Obj + \sqrt{T_{Obj} + T_{Lnk}}} = \frac{15+6}{1+2+3} = 3{,}5$$

То есть численная оценка для данной диаграммы равна 3,5.

Контрольные вопросы:

1. Основное назначение UML-моделей?
2. Каково назначение элемента актер в диаграммах прецедентов?
3. Количественная оценка вашей диаграммы, сделайте вывод о правильности и наглядности ее построения.
4. Какие среды для создания диаграмм вы еще знаете?

4. Методические указания для самостоятельной работы студентов

1) Систематическая проработка конспектов занятий, учебной и специальной технической литературы.

Методические рекомендации:

Внимательно прочитайте теоретический материал – конспект, составленный на лекционном занятии, рекомендованную лектором учебную литературу, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.).

2) Написание доклада; подготовка к сообщению на занятии по заданной преподавателем теме (с использованием Интернет ресурсов).

Методические рекомендации:

Доклад в отличии от реферата отражает одну точку зрения на проблему, не предполагает ее исследования в сравнении и анализе.

1. Выберете тему из предложенной преподавателем тематики докладов и сообщений. Вы можете самостоятельно предложить тему с учетом изучаемого теоретического материала.

2. При подготовке доклада, сообщения используйте техническую литературу по выбранной теме, электронные библиотеки или другие Интернет-ресурсы.

3. Сделайте цитаты из книг и статей по выбранной теме (обратите внимание на непонятные слова и выражения, уточнить их значение в справочной литературе).

4. Проанализируйте собранный материал и составьте план сообщения или доклада, акцентируя внимание на наиболее важных моментах.

5. Напишите основные положения сообщения или доклада в соответствии с планом, выписывая по каждому пункту несколько предложений.

6. Перескажите текст сообщения или доклада, корректируя последовательность изложения материала.

7. Подготовленный доклад должен сопровождаться презентацией, иллюстрирующей его основные положения.

Показатели оценки результатов самостоятельной работы студента (СРС):

– полнота и качественность информации по заданной теме;

– свободное владение материалом сообщения или доклада;

– логичность и четкость изложения материала;

– наличие и качество презентационного материала.

3) Подготовка к контрольным работам по разделам дисциплины.

Методические рекомендации:

Следует внимательно прочитать материал по конспекту, составленному на учебном занятии, затем прочитать его по учебному пособию. По возможности необходимо разобраться с непонятным, в частности новыми терминами, и ответить на контрольные вопросы для самопроверки.

Освоив теоретический материал, можно приступить к выполнению заданий.

5. Методические указания для выполнения курсовой работы

Курсовая работа по дисциплине “Проектирование АСОИУ” направлена на применение теоретических знаний и навыков практической работы с инструментальными средами, полученных в процессе изучения курса. Работа призвана способствовать формированию у студентов инженерных навыков проектирования.

Курсовая работа является составной частью заключительного этапа подготовки бакалавра и имеет следующие основные цели:

- систематизация, закрепление и обобщение теоретических и практических знаний по профилю и применение этих знаний для решения конкретных научных, технических, экономических и производственных задач;

- умение разрабатывать актуальные инженерно-технические проблемы, имеющие важное теоретическое и практическое значение;

- развитие навыков ведения самостоятельной работы и овладения методами исследования экспериментирования и современной информационной технологией при решении разрабатываемых в курсовой работе проблем и задач;

- выявление уровня знаний и подготовленность студентов для самостоятельной работы в условиях рыночной экономики, прогресса науки, техники и информационных технологий.

- анализ организационных систем, рассматриваемых в качестве объекта автоматизации и объектно-ориентированный анализ, и проектирование информационных систем.

Результаты выполненной работы представляются в виде курсовой работы, включающей пояснительную записку и мультимедийные разработки студента.

Рекомендуется тему курсовой работы выбирать согласно теме дипломного проекта или проводимых студентом учебно-исследовательских работ. Такой подход позволяет добиться глубокого понимания студентом предметной области и положительно сказывается на качестве выполнения курсовой работы. Для студентов, не определившихся с дипломным проектом, приведен перечень тем.

5.1. Организация работы

5.1.1. Выбор темы курсовой работы

Выполнение курсовой работы должно проводиться, как правило, по материалам тех предприятий и организаций, которые являются местами будущей работы студентов после окончания ими университета.

Темы курсовых работ определяются выпускающей кафедрой с учетом специфики мест практики и в соответствии с направлением и профилем подготовки студента.

Студент может предложить свою тему с обоснованием целесообразности ее разработки. При выборе темы курсовой работы учитываются индивидуальные особенности студента, его знания, умения и навыки в той или иной области работ по проектированию АСОИУ, технологии использования и оценки АС, наличие конкретных материалов, полученных в результате практики, способность самостоятельно работать с научно-технической и справочной литературой.

Тема курсовой работы должна быть направлена на решение реальной задачи автоматизации организационно-административного управления, а также управления технологическими или экономическими процессами с использованием ЭВМ, современных информационных технологий и интеллектуальных инструментальных средств. Объектом проектирования является любой объект управления, требующий автоматизации процесса управления. Объект исследования выбирается самостоятельно студентом и утверждается преподавателем.

Примеры тем курсовых работ по профилю «Автоматизированные системы обработки информации и управления»:

1. «Автоматизированная система управления проектами и документооборота».
2. «Система сбора и передачи данных для компаний по перевозке грузов».
3. «Создание интегрированной среды разработки электронных учебников на основе технологии XML».
4. «Разработка инструментальных средств дистанционного обучения на основе технологии Web CT».
5. «АСУ сети мебельных магазинов»
6. «Автоматизированная система склада крупного предприятия»
7. «Разработка подсистемы «ОСАГО АИС страховой компании».
8. Разработка информационной системы организации «Юношеская Лига восточных боевых единоборств
9. «Автоматизированный учет и контроль движения товаров на складе».
10. «Разработка автоматизированной системы маркетинговых исследований малого бизнеса».
11. «Автоматизированная система финансовой отчетности».
12. Разработка информационной системы МП «Музыкальные инструменты».
13. «Проектирование информационного обеспечения мониторинга сбыта нефтепродуктов» ОАО «АНК».
14. Подсистема «Управление движением» АСУ ГУП Гортранс.
15. Разработка автоматизированной системы сбора и обработки оперативной информации ОВД.
16. Разработка АИС типографии.
17. АИС терапевтического отделения клинической больницы.
18. Централизованная АСУ «Кафе».
19. «АСУ складским комплексом сети компьютерных магазинов».
20. «Разработка информационной системы среднего образовательного учреждения».

Приложение должно включать распечатку программы решения проектируемой задачи и дискету с текстом курсового проекта и программой.

Графический материал должен иллюстрировать основное содержание курсовой работы и включать следующие чертежи, выполненные в виде презентаций:

1. Функциональная структура системы.
2. Блок-схема алгоритма задачи и ее модулей.
3. Результаты решения задачи (таблицы, графики и др.).

5.1.2. Этапы выполнения курсовой работы

Курсовая работа должна состоять из пояснительной записки, графического материала и программного обеспечения. Курсовая работа, как правило, должна являться составной частью выпускной квалификационной работы. Дальнейшая доработка темы должна осуществляться в ходе выполнения выпускной квалификационной работы.

Курсовая работа включает в себя постановку задачи с представлением предметной области задачи проектирования, анализ существующих или возможных решений поставленной задачи с кратким обзором литературных источников, алгоритмическую проработку решений, выбор среды реализации с использованием средств автоматизации проектирования, как правило, CASE-инструмент.

В курсовой работе, как правило, должны быть представлены результаты отладки проектируемых компонент, в средствах выбранной CASE-среды.

Целесообразно выделить три основных части курсовой работы:

– постановка задачи, анализ решений и функциональная разработка системы;

– разработка информационного обеспечения, функциональных модулей, интерфейсов;

– отладочные работы, оформление записки, графических материалов и подготовка к защите.

Структура курсовой работы:

		Объем в стр. не менее
1.	Аннотация (на русском и английском языках)	1
2.	Введение	1
3.	Формулировка темы	1
4.	Анализ аналогичных решений на основании обзора литературы и обоснование выбранного решения	3
5.	Конструктивно-технологические или теоретические решения, на которых построен проект	12
6.	Результаты проектирования	25
7.	Заключение	1
8.	Список используемой литературы	1
	ПРИЛОЖЕНИЕ	

Предпроектная стадия курсовой работы, включающая проведение необходимых исследований, работ по формированию структуры АСОИУ и получении, в конечном счете, технического задания на проектирование может проходить по двум принципиально различным вариантам.

Первый – классический фундаментальный подход, связан с проведением исследований по вскрытию законов, на основе которых функционирует исследуемая система об-

работки информации и управления. Далее, вследствие выявленных фундаментальных законов, строится формальная модель, связывающая входы, выходы системы, влияние внешних воздействий на нее. Таким образом, получаем формально-математическое представление системы, которое и может в дальнейшем служить основой для функционального, инфологического и техно-рабочего проектирования АСОИУ.

Второй подход, достаточно часто применяемый при построении АСОИУ, включает в себя проведение информационного обследования объекта (предметной области), выявление основных информационных потоков, построения, как правило, имитационной модели функционирования объекта и далее выход также на инфологическое и техно-рабочее проектирование.

Первый подход в большей степени гарантирует получение высококачественной разработки, но требует больших интеллектуальных и финансовых затрат. Второй подход обеспечивает, на первый взгляд, меньшие затраты, но вероятней всего за счет неудачных и малоэффективных решений, общие затраты при реализации работ по второму варианту могут быть и значительно больше, чем в первом случае.

Однако выбор схемы решения всегда остается за разработчиком.

Анализ является первым этапом создания АСОИУ, на котором требования заказчика уточняются, формализуются и документируются. Фактически на этом этапе дается ответ на вопрос: «Что должна делать будущая система?». Именно здесь лежит ключ к успеху всего проекта.

Структурный анализ начинается с исследования того, как организована система управления предприятием, с обследования функциональной и информационной структуры системы управления. По результатам обследования аналитик на первой стадии анализа строит обобщенную логическую модель исходной предметной области, отображающую ее функциональную структуру, особенности основной деятельности и информационное пространство, в котором эта деятельность осуществляется. Используя специальную терминологию, можно сказать, что аналитик строит модель «как есть».

Вторая стадия работы, к которой привлекаются заинтересованные представители заказчика, а при необходимости и независимые эксперты, состоит в анализе модели «как есть», выявлении ее недостатков и узких мест, определение путей совершенствования системы управления на основе выделенных критериев качества.

Третья стадия анализа, содержащая элементы проектирования, – создание усовершенствованной обобщенной логической модели, отображающей реорганизованную предметную область или ее часть, которая подлежит автоматизации. Эту модель можно назвать моделью «как надо», т.е. здесь происходит формализация системы.

Наряду со структурным подходом существует и более мощный подход называемый объектно-ориентированным ООП. Эта методология создана для проектирования больших и сложных систем и имеет ряд преимуществ перед структурным подходом.

ООП базируется на создании объектной модели, которая описывает структуру объектов, составляющих систему, их атрибуты, операции, взаимосвязи с другими объектами. Цель разработки объектной модели – описать объекты, составляющие в совокупности проектируемую систему, а также выявить и указать различные зависимости между объектами. *Декомпозиция проблемы на объекты* – творческий, плохо формализуемый процесс.

В объектной модели должны быть отражены те понятия и объекты реального мира, которые важны для разрабатываемой системы. В объектной модели отражается, прежде всего, прагматика разрабатываемой системы, что выражается в использовании терминологии прикладной области, связанной с использованием разрабатываемой системы.

Методы, которые используются для изучения предметной области и технологии получения от экспертов сведений о системе, подлежащей описанию, называются сбором данных, а в информатике она больше известна как опрос (интервьюирование) или извлечение знаний. Но как бы она не называлась, способность собрать необходимую информацию, основанную на знаниях экспертов, весьма существенна для построения точной и по-

лезной модели. Поэтому технология сбора информации составляет важную часть структурной методологии SADT.

Следует учитывать следующие особенности автоматизированных систем как объектов стандартизации:

1) техническое задание является основным документом в соответствии, с которым проводят создание АСОИУ и приемку его заказчиком,

2) АСОИУ, как правило, создают проектным путем с комплектацией изделиями серийного и единичного производства и проведением строительных, монтажных, наладочных и пусковых работ, необходимых для ввода в действие системы;

3) в общем случае АСОИУ состоит из программно-технических (ПТК), программно-методических комплексов (ПМК) и компонентов технического, программного и информационного обеспечения. Компоненты этих видов обеспечения, а также ПМК и ПТК должны изготовляться, и поставляется как продукция производственно-технического назначения. Компоненты могут входить в систему в качестве самостоятельных частей или могут быть объединены в комплексы;

4) создание АСОИУ в организациях (предприятиях) требует специальной подготовки пользователей и обслуживающего персонала системы;

5) функционирование АСОИУ и комплексов обеспечивается совокупностью организационно-методических документов, рассматриваемых в процессе создания как компоненты правового, методического, лингвистического, математического, организационного и др. видов обеспечения. Отдельные решения, получаемые в процессе разработки этого обеспечения, могут реализовываться в виде компонентов технического, программного или информационного обеспечения;

6) совместное функционирование и взаимодействие различных систем и комплексов осуществляется на базе локальных сетей ЭВМ.

Спецификации и соглашения, принятые для локальных сетей ЭВМ, обязательны для обеспечения совместимости систем, комплексов и компонентов.

Стандартизация в области АСОИУ является составной частью работ по обобщенной проблеме «Информационная технология».

Объем пояснительной записки (не включая Приложения) не менее 45 страниц.

Текст пояснительной записки должен быть грамотно и аккуратно напечатан студентом на ЭВМ на листах формата А4 с соблюдением требований, предъявленных к текстовой документации (ГОСТ 2.105-79 и 2.106-68).

Текстовая часть должна быть дополнена иллюстрациями в виде структурных, функциональных и принципиальных схем, блок-схем алгоритмов и программ, графиками и др.

Все иллюстрации должны именоваться рисунками, выполняться на листах белой бумаги форматов А4 (210х297 мм) или А2 (297 х420 мм) - ГОСТ 2.301-68 [2], иметь сквозную нумерацию и наименование (применение миллиметровки разрешается только для построения экспериментально полученных графиков).

Листы с рисунками необходимо выполнять на отдельных листах и располагать непосредственно за листом текста, в котором имеется первая ссылка на этот рисунок. Ссылка в тексте на графический материал, выполненный на листах для защиты проекта, не разрешается.

Таблицы, включаемые в пояснительную записку, должны иметь нумерацию и заголовки.

Не допускается переписывание студентом в пояснительную записку каких-либо разделов из книг, журналов или отчетов.

В тексте должно соблюдаться единообразие терминов и обозначений, не допускается сокращение слов, кроме общепринятых.

В соответствии с Единой системой программной документации (ЕСПД) ГОСТ 19.402-78 (СТСЭВ 2092-80), «Описание программы» в курсовой работе в разделе «Описание программного обеспечения задачи» должны быть указаны:

обозначение и наименование программы, программное обеспечение, необходимое для функционирования программы, языки программирования, на которых написана программа;

классы решаемых задач и (или) назначение программы и сведения о функциональных ограничениях на применение;

приведена структура программы с описанием функций составных частей и связей между ними;

связи программы с другими программами;

характер, организация и предварительная подготовка входных данных формат, описание и способ кодирования входных данных;

характер и организация выходных данных, формат, описание и способ кодирования выходных данных.

В соответствии с ЕСПД ГОСТ 19.505-79 «Руководство оператора». «Требования к содержанию и оформлению» в курсовой работе в разделе «Инструкция пользователю» должны быть указаны:

сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации;

указаны условия, необходимые для выполнения программы (минимальный и (или) максимальный состав аппаратурных и программных средств и т.п.);

последовательность действий пользователя, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых пользователь осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды;

должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия пользователя (действия пользователя в случае сбоя, возможности повторного запуска программы и т.п.).

В списке литературы должны быть указаны все использованные источники и приведены следующие сведения:

- для книг - фамилии и инициалы авторов, название, издательство, место и год выпуска;
- для статей - фамилии и инициалы авторов, название статьи, журнала или сборника, год издания, том, номер, стр.;
- для отчетов - название, организация, место и год выпуска.

При ссылке в тексте на источник приводится порядковый номер использованной литературы, заключенный в квадратные скобки.

В приложение включает распечатки программы.

5.1.3. Графический материал

Графический материал работы должен состоять из чертежей, выполненных в соответствии с требованиями ЕСКД, ЕСПД и других действующих ГОСТов. Чертежи должны быть выполнены в мультимедийном исполнении и снабжены штампом. Каждый чертеж должен быть снабжен основной надписью, графы которой должны быть заполнены в соответствии с образцом.

Макеты чертежей перед окончательным оформлением необходимо показать руководителю проекта.

К основным ГОСТам, которые необходимо использовать при оформлении курсовой работы, относятся:

ГОСТ 34.201-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем.

ГОСТ 34.602-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. С. 106-116.

РД 50-682-89 Методические указания. Информационная технология Комплекс стандартов и руководящих документов на автоматизированные системы. Общие положения. С. 157-161.

РД 50-680-88 Методические указания. Автоматизированные системы. Основные положения. С. 152-156.

ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. С. 100-105.

ГОСТ 34.401-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Средства технические периферийные автоматизированных систем. Типы и технические требования.

РД 50-34.698-90 Методические указания. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов. С. 127-151.

ГОСТ 34.003-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения.

Р50-34.119-90 Рекомендации. Информационная технология. Комплекс стандартов на автоматизированные системы. Архитектура локальных вычислительных сетей в системах промышленной автоматизации. Общие положения.

ГОСТ 24.301-80* Система технической документации на АСУ. Общие требования к выполнению текстовых документов.

ГОСТ 24.302-80 Система технической документации на АСУ. Общие требования к выполнению схем. С. 22-24.

ГОСТ 24.303-80 Система технической документации на АСУ. Обозначения условные графические технических средств. С.25-31.

ГОСТ 24.304-80 Система технической документации на АСУ. Требования к выполнению чертежей С.32-34.

ГОСТ 19.401-78* Единая система программной документации. Текст программы. Требования к содержанию и оформлению.

ГОСТ 19.402-78* Единая система программной документации Описание программы.

Кроме того, графический материал может быть представлен в виде и мультимедийных разработок.

5.2. Защита курсовой работы

Курсовая работа, подписанная исполнителем, представляется на проверку руководителю за три дня до назначенного срока защиты.

Защита курсовой работы проводится индивидуально каждым студентом на заседании комиссии, состав которой утверждается кафедрой, с обязательным участием руководителя курсовой работы.

Студент, выполнивший курсовую работу, делает доклад (5-7 минут) и отвечает на вопросы комиссии.

В докладе студенту необходимо изложить важнейшие этапы, особенности и результаты работы, не углубляясь в тонкости конкретных технических решений, четко сформулировать конечные результаты.

Вопросы, задаваемые студенту, могут касаться деталей выполненной работы, а также разделов курсов, по которым выполнялась работа.

Результаты защиты курсовой работы определяются оценками "отлично", "хорошо", "удовлетворительно", "неудовлетворительно", которые проставляются на титульном листе пояснительной записки. При оценке работы учитывается качество выполнения и оформления курсовой работы, уровень защиты курсовой работы и ответов на вопросы, мнение руководителя.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Основная:

Золотов С.Ю. Проектирование информационных систем [Электронный ресурс]: учебное пособие/ Золотов С.Ю.— Электрон. текстовые данные.— Томск: Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2013.— 88 с.— Режим доступа: http://www.iprbookshop.ru/13965.— ЭБС «IPRbooks», по паролю

Золотов С.Ю. Проектирование информационных систем [Электронный ресурс] : учебное пособие / С.Ю. Золотов. — Электрон. текстовые данные. — Томск: Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2013. — 88 с. — 978-5-4332-0083-8. — Режим доступа: http://www.iprbookshop.ru/13965.html

Дополнительная:

Кандаурова Н.В. Технологии обработки информации [Электронный ресурс]: учебное пособие/ Н.В. Кандаурова, В.С. Чеканов— Электрон. текстовые данные.— Ставрополь: Северо-Кавказский федеральный университет, 2014.— 175 с.— Режим доступа: http://www.iprbookshop.ru/63145.html.— ЭБС «IPRbooks», по паролю

Синицын Ю.И. Волоконно-оптические линии связи в компьютерных сетях и телекоммуникациях [Электронный ресурс]: методические указания к практическим и лабораторным занятиям/ Синицын Ю.И.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2014.— 142 с.— Режим доступа: http://www.iprbookshop.ru/50050.— ЭБС «IPRbooks», по паролю

Учебно-методическое пособие и задания на курсовую работу по дисциплине Технологии обработки информации [Электронный ресурс]/ — Электрон. текстовые данные.— М.: Московский технический университет связи и информатики, 2016.— 26 с.— Режим доступа: http://www.iprbookshop.ru/61564.html.— ЭБС «IPRbooks», по паролю

Алексеев В.А. Основы проектирования и реализации баз данных [Электронный ресурс]: методические указания к проведению лабораторных работ по курсу «Базы данных»/ В.А. Алексеев— Электрон. текстовые данные.— Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2014.— 26 с.— Режим доступа: http://www.iprbookshop.ru/55122.html.— ЭБС «IPRbooks»

в) программное обеспечение и Интернет-ресурсы:

Перечень программного обеспечения

№	Перечень программного обеспечения (обеспеченного лицензией)	Реквизиты подтверждающих документов
1	Операционная система MS Windows 7 Pro	DreamSpark Premium Electronic Software Delivery (3 years) договору – Сублицензионный договор №Tr000074357/KHB 17 от 01 марта 2016 года

№	Перечень программного обеспечения (обеспеченного лицензией)	Реквизиты подтверждающих документов
2	Операционная система MS Windows 10 Education	DreamSpark Premium Electronic Software Delivery (3 years) договору – Сублицензионный договор №Tr000074357/KHB 17 от 01 марта 2016 года
3	MS office 2010 standard	Лицензия Microsoft office 2010 Standard RUS OLM ML Academic 50,договор №492 от 28 июня 2012 года
4	MS access 2010	DreamSpark Premium Electronic Software Delivery (3 years) договору – Сублицензионный договор №Tr000074357/KHB 17 от 01 марта 2016 года
5	Автоматизированная информационная библиотечная система «ИРБИС 64»	Лицензия коммерческая по договору №945 от 28 ноября 2011 года
6	Программный комплекс «Консультант Плюс»	Лицензия коммерческая по договору №21 от 29 января 2015 года
7	7-Zip	Бесплатное распространение по лицензии GNU LGPL http://www.7-zip.org/licese.txt
8	LibreOffice	Бесплатное распространение по лицензии Mozilla Public Licence Version 2.0
9	Ramus	Бесплатное распространение по лицензии GNU GPL https://www.gnu.org/licenses/quick-guide-gplv3.ru.html
10	Open ModelSphere	Бесплатное распространение по лицензии EPL https://www.eclipse.org/legal/epl-v10.html
12	Argo UML	Бесплатное распространение по лицензии EPL https://www.eclipse.org/legal/epl-v10.html
13	Open System Architect	Бесплатное распространение по лицензии GNU GPL http://www.gnu.org/licenses/old-licenses/gpl-2.0.htm
14	Umlet	Бесплатное распространение по лицензии

№	Перечень программного обеспечения (обеспеченного лицензией)	Реквизиты подтверждающих документов
		GNU GPL http://www.gnu.org/licenses/old-licenses/gpl-2.0.htm

Перечень Интернет-ресурсов:

	Наименование ресурса	Краткая характеристика
	amursu.ru	Сайт ФГБОУ ВПО АмГУ
	Электронная библиотечная система www.iprbookshop.ru	ЭБС IPRbooks — научно-образовательный ресурс для решения задач обучения в России и за рубежом. Уникальная платформа ЭБС IPRbooks объединяет новейшие информационные технологии и учебную лицензионную литературу. Контент ЭБС IPRbooks отвечает требованиям стандартов высшей школы, СПО, дополнительного и дистанционного образования. ЭБС IPRbooks в полном объеме соответствует требованиям законодательства РФ в сфере образования.
	http://elibrary.ru	Научная электронная библиотека eLIBRARY.RU. Крупнейший российский информационный портал в области науки, технологии, медицины и образования, содержащий рефераты и полные тексты более 12 млн научных статей и публикаций. На платформе eLIBRARY.RU доступны электронные версии более 1400 российских научно-технических журналов, в том числе более 500 журналов в открытом доступе.