Разработка имитационной модели программного обеспечения информационной системы "Центр обслуживания абонентов" диплом 2010 по информатике

Информатика

55 pag.

Разработка имитационной модели программного обеспечения информационной системы "Центр обслуживания абонентов"

Содержание

Введение

Сегодня нет такого вида электросвязи, который развивался бы столь быстро, как беспроводная радиотелефония, и особенно сотовая связь, обслуживающая подвижных пользователей. Число таких пользователей ежегодно увеличивается где-то на 40% и недавно уверенно перешагнуло 350-миллионный рубеж. Следует заметить, что отдельные разновидности сотовых сетей подвижной связи показывают еще более впечатляющие темпы роста.

С момента, когда сотовые сети вышли из стадии научно-технических разработок на коммерческий рынок (рубеж 70-80-х годов), прошло не так уж много времени, сотовый телефон ныне перестал быть символом престижа и стал инструментом, позволяющим более эффективно использовать рабочее время, оперативно управлять технологическими, экономическими и другими процессами. Сотовые сети при этом не только растут вширь, но и неуклонно наращивают количество предоставляемых дополнительных услуг.

Если взять плотность сотовых сетей, которой достигли многие развитые страны, то бесспорным лидером среди них является Финляндия, где около 70% населения имеют сотовые телефоны. Немного отстают от нее соседи по Скандинавии (от 50 до 60%), между которыми вклинился кусочек КНР (бывший Гонконг). Ещё шесть стран, включая Австралию и Японию, перешагнули 30% -ный рубеж, а целая группа стран, в том числе Великобритания и США, вплотную к нему приблизилась. Примерно 20% -ные показатели имеют Германия, Франция, Испания, Канада и ряд других стран. Впрочем, можно еще долго перечислять успехи подвижной связи, но и уже приведенной информации достаточно, чтобы сделать некоторые выводы.

Все сказанное выше свидетельствует о том, что в большинстве этих стран плотность радиотелефонных линий уже достигла уровня плотности линий традиционной телефонии (примерно 50-60%) или составила весьма обширные территории, либо очень многочисленные население, либо и то, и другое одновременно. Поэтому даже 5% в Бразилии и 3% в КНР без учета Гонконга (какая маленькая цифра и одновременно чудовищно большое число

пользователей) оказывается весомее, чем, например, 27% где-нибудь в Голландии. В общем, сотовая связь уже стала продуктом массового потребления и продолжает наращивать темпы роста.

Возможно, через несколько лет мы станем свидетелями того, как группы пользователей традиционными и сотовыми телефонами практически сравняется по численности. Кстати, сегодня 1 млрд проводных телефонных линий. Поскольку уже есть достаточно оснований сомневаться в привычном первенстве проводной телефонной связи, зарубежные аналитики начинают рассматривать сотовую связь как услугу, равную проводной, а некоторые даже отдают ей первенство.

Бурно развивающийся мировой рынок услуг сотовой связи привлекает внимание многих компаний. Например, только в России в этот бизнес вовлечено около двухсот компаний - операторов. Все убыстряющиеся темпы развития общества настоятельно требуют получения более персональных услуг, а именно: возможности телефонного разговора (и не только) с любым абонентом, в любом месте и в любое время.

Глава 1. Имитационное моделирование деятельности "Центра обслуживания абонентов"

Сотовая связь - это разновидность высокоподвижной радиосвязи, отличающаяся прежде всего массовостью обслуживания абонентов на ограниченной территории. При этом исторически сложилось так, что сотовая связь постепенно расширяла сферу обслуживания телефонной сети общего пользования (ТфОП). Вообще-то, сам термин "сотовая связь - это общепринятое сокращенное наименование услуги, получаемой с помощью развернутых сотовых сетей подвижной связи, выполненных на базе соответствующих систем. Таким образом, этот термин характеризует именно подвижную связь.

В настоящее время высшим приоритетом операторов сетей подвижной связи является предоставление новых услуг абонентам, призванных облегчить процесс эксплуатации, методы ведения связи, предоставить дополнительный информационный сервис, упростить процедуру оповещения и взаиморасчетов.

К тому же наблюдаемое в последнее время быстрое увеличение количество абонентов заставляет операторов внедрять новейшие системы автоматического обслуживания, тем самым уменьшая затраты. Применительно к абонентам услуги, предоставляемые телефонным сервером, помогут превратить его сотовый телефон в автоматического секретаря.

Десятки операторов по всей России предлагают своим абонентам одни из самых современных видов сотовой связи. Компании ведут отчаянную конкурентную борьбу между собой за каждого клиента.

В данной дипломной работе требуется разработать имитационную модель программного обеспечения информационной системы "Центр обслуживания абонентов" (далее просто Системы).

Перед современными предприятиями часто встает задача оптимизации технологических процессов. Широко известный метод функционального моделирования позволяет обследовать существующие бизнес-процессы,

выявить их недостатки и построить идеальную модель деятельности предприятия. Построение функциональной модели осуществляется от общего к частному - сначала описывается общая схема деятельности предприятия, затем шаг за шагом все более и более подробно описываются конкретные технологические процессы. Такой подход весьма эффективен, однако на уровне наибольшей детализации, когда рассматриваются конкретные технологические операции, для оптимизации этих операций функциональной модели может оказаться недостаточно. В этом случае целесообразно использовать имитационное моделирование.

Имитационное моделирование - это метод, позволяющий строить модели, учитывающие время выполнения функций. Полученную модель можно “проиграть” во времени и получить статистику происходящих процессов так, как это было бы в реальности. В имитационной модели изменения процессов и данных ассоциируются с событиями. “Проигрывание" модели заключается в последовательном переходе от одного события к другому. Обычно имитационные модели строятся для поиска оптимального решения в условиях ограничения по ресурсам, когда другие математические модели оказываются слишком сложными.

Прежде всего, мне представляется целесообразным дать общую характеристику предприятия, применительно к которой я буду строить имитационную модель и кратко остановлюсь на особенностях задач управления этого предприятия.

По виду деятельности Центр обслуживания абонентов предоставляет услуги связи абонентам и услуги по их дальнейшему обслуживанию, по масштабу считается малым предприятием.

Моя задача разработать модель качественной системы, в которой информация едина и достоверна, причём очень важно, что достоверность информации гарантируется собственно системой управления в целом, а не отдельными людьми. В результате достигается возможность простого и эффективного контроля за работой компании в целом, контроля отдельных процессов и даже контроля деятельности отдельных сотрудников.

Система "Центр обслуживания абонентов" предназначена для предоставления услуг связи абонентам. Система позволяет операторам сотовой связи создать электронные аналоги договоров об оказании услуг связи, вести учет всех абонентов и в нужное время обслужить клиента, предоставив ему необходимую информацию. Электронный договор имеет юридическую силу аналогичную бумажному варианту подписанного договора. В результате, операторы избавлены от необходимости постоянного ведения бумажной работы и хранения "твердых" копий документов (копия документа хранится у абонента по желанию).

Система состоит из двух основных компонентов: серверной части программного обеспечения и клиентской части (с ней работают операторы), взаимодействующих по принципу трехзвенной архитектуры. Клиентская часть предназначена для взаимодействия с операторами. Для начала работы с системой оператору необходимо пройти процедуру регистрации, выбрать оператора связи с которым будет осуществляться работа, производится сеанс связи с Центральным офисом, в процессе которого эти данные передаются на сервер центрального офиса. После этого, оператор имеет возможность начать работу с Системой.

Типичный сценарий работы оператора с Системой выглядит следующим образом. Оператор открывает программу и выбирает одного из имеющихся операторов сотовой связи, так как у каждого оператора свой Центральный офис и сервер. Электронные документы абонентов бывают следующих видов: договор об оказании услуг связи, заявления для замены sim-карты, замены абонентского номера, на детализацию счета, пополнения счета. При создании договора или заявления открывается окно формы соответствующего документа, оператор заполняет все необходимые поля формы, сохраняет документ. При заполнении реквизитов документов и сохранением документа выполняется проверка правильности заполнения реквизитов документов.

Для передачи документов в центр необходимо провести сеанс связи с сервером центра в процессе которого документы, созданные оператором

передаются на сервер, а уже хранящиеся документы клиента, находящиеся на сервере центра передаются на компьютер оператора. Этот процесс называется синхронизацией. После проведения синхронизации оператор имеет возможность просмотреть детализацию счета за любой произвольный период времени. При этом система формирует перечень всех звонков, произведенных абонентом за просматриваемый период с указанием длительности разговора. Перечень звонков из списка может быть распечатан по форме, соответствующей типу этого документа. В момент проведения синхронизации система запрашивает имя абонента для поиска его в имеющейся БД и только потом осуществляет сеанс связи. Если введенный абонент отсутствует, система об этом сообщает.

1.1 Диаграммы потоков данных

Укрупнено методы построения моделей предприятий можно разделить на структурные и объектно-ориентированные. Каждая из этих групп методов включает в себя несколько вариантов конкретных методик. Структурные методы на сегодняшний день имеют наибольшее распространение, поэтому их мы рассмотрим в первую очередь.

Структурным принято называть такой метод исследования системы или процесса, который начинается с общего обзора объекта исследования, а затем предполагает его последовательную детализацию.

Структурные методы имеют три основные особенности:

расчленение сложной системы на части, представляемые как "черные ящики", а каждый черный ящик реализует определенную функцию системы управления;

иерархическое упорядочение выделенных элементов системы с определением взаимосвязей между ними;

использование графического представления взаимосвязей элементов системы;

Модель, построенная с применением структурных методов представляет собой иерархический набор диаграмм, графически изображающих выполняемые системой функции и взаимосвязи между ними. Это рисунки, на которых показан набор прямоугольников, определенным образом связанных между собой. В диаграммы также включается текстовая информация для обеспечения точного определения содержания функций и взаимосвязей. Использование графического представления процессов существенно повышает наглядность модели и облегчает процесс ее восприятия. От обычных рисунков, с помощью которых можно представить процесс управления, структурные диаграммы отличаются тем, что выполняются по вполне определенным правилам, а процесс их составления и анализа поддерживается соответствующим программным обеспечением.

За последнее десятилетие сформировалось новое направление в программотехнике - **CASE** (**C**omputer-**A**ided **S**oftware/**S**ystem **E**ngineering). В настоящее время не существует общепринятого определения CASE. Содержание этого понятия обычно определяется перечнем задач, решаемых с помощью CASE, а также совокупностью применяемых методов и средств. Очень грубо, CASE - технология представляет собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных систем программного обеспечения (ПО), поддержанную комплексом взаимоувязанных средств автоматизации. CASE - это инструментарий для системных аналитиков, разработчиков и программистов, заменяющий им бумагу и карандаш на компьютер для автоматизации процесса проектирования и разработки ПО.

В составе методологий структурного анализа к наиболее распространенным можно отнести следующие:

SADT (Structured Analysis and Design Technique) - технология структурного анализа и проектирования и ее подмножество стандарт IDEF0;

DFD (Data Flow Diagrams) - диаграммы потоков данных;

ERD (Entity-Relationship Diagrams) - диаграммы "сущность-связь";

STD (State Transition Diagrams) - диаграммы переходов состояний.

В моей работе была использована методология DFD (Data Flow Diagrams). В DFD методологии исследуемый процесс разбивается на подпроцессы и представляется в виде сети, связанной потоками данных. В число элементов данной методолошии входят процессы, потоки данных и хранилища. Хранилище позволяет в необходимых случаях определить данные, которые будут сохраняться в памяти между процессами.

Диаграммы потоков данных (DFD - Data Flow Diagram) являются основным средством моделирования функциональных требований проектируемой системы. С их помощью эти требования разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств - продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Декомпозиция DFD осуществляется на основе процессов: каждый процесс может раскрываться с помощью DFD нижнего уровня. Важную специфическую роль в модели играет специальный вид DFD - контекстная диаграмма, моделирующая систему наиболее общим образом. Контекстная диаграмма отражает интерфейс системы с внешним миром, а именно, информационные потоки между системой и внешними сущностями, с которыми она должна быть связана.

Индивидуальные данные в системе часто являются независимыми. Однако иногда необходимо иметь дело с несколькими независимыми данными одновременно. Для этого используется диаграммы декомпозиции. Применение этих операций над данными позволяет обеспечить структуризацию данных, увеличивает наглядность и читабельность диаграмм.

Построю контекстную DFD диаграмму системы. Для этого необходимо изобразить основную функцию рассматриваемой системы "Центр обслуживания абонентов" и внешние по отношению к ней сущности, а также взаимосвязи между внешними сущностями и функцией системы. Контекстная диаграмма будет выглядеть следующим образом.

Рис.1. Контекстная диаграмма системы "Обслуживание абонентов"

Основным процессом является *Обслуживание абонентов*, т.к вся деятельность системы направлена именно на это. Внешними сущностями являются *Оператор* и *Клиент*. На первоначальном этапе взаимодействия, то есть, для заключения Договора, клиент предоставляет документы, составляет анкету, после чего она рассматривается оператором. В случае положительного ответа заключается договор об оказаниях услуг связи, иначе клиенту сообщается об отказе.

Далее клиент в случае необходимости посещает центр для заполнения заявления. Заявление может быть по разному поводу (просьба детализации счета, замена абонентского номера, замена sim-карты, заявка на пополнение счета…). Оператором рассматриваются заявления, и выносится решение об удовлетворении либо об отказе, что показано на диаграмме.

После создания контекстной диаграммы, я постаралась рассмотреть функции системы более подробно и построить диаграмму детализации первого уровня. Построенная диаграмма первого уровня также имеет множество процессов, которые в свою очередь могут быть декомпозированы в DFD нижнего уровня. Таким образом строится иерархия DFD с контекстной диаграммой в корне дерева. Этот процесс декомпозиции продолжается до тех пор, пока процессы могут быть эффективно описаны с помощью коротких (до одной страницы) миниспецификаций обработки (спецификаций процессов).

Рис.2. Диаграмма детализации первого уровня системы "Обслуживание абонентов"

Обслуживание абонентов может быть представлено в виде семи основных функций:

Рассмотрение анкеты

Заключение договора

Отказ от заключения договора

Подключение к сети

Рассмотрение заявления

Отказ от исполнения заявления

Исполнение заявления

Целью каждой функции является учет выполняемых в рамках данного процесса действий и отражение их результата в проектируемой информационной системе. Коротко прокомментирую каждую из них.

Рассмотрение анкеты.

Данная операция осуществляется оператором и не может быть полностью автоматизирована, так как изучение анкеты проходит в два этапа, второй из которых заключается в обязательным заполнении анкеты вручную абонентом. Результатом данной операции является решение о внесение в систему информации о новом абоненте. Отсутствие ее в общей модели привело бы к выпадению одного из звеньев цепи обслуживания абонента.

Заключение договора.

Результатом данной операции является внесение в систему информации о клиенте.

Отказ от заключения договора.

Оператор решает в силу тех или иных причин отказать в заключении договора о предоставлении услуг связи.

Подключение к сети.

При заключении договора необходимо выбрать оператора сети, приобрести sim-карту и соответственно абонентский номер.

Рассмотрение заявления.

В процессе пользования услугами связи у абонента могут возникнуть те или иные требования, которые ему необходимо отразить в заявлении. Заявление рассматривается операторами.

Отказ от исполнения заявления.

При наличии новых заявлений оператор осуществляет их проверку. Если принято решение о нецелесообразности исполнения данного требования, то сообщается абоненту об отказе.

Исполнение заявления

В случае принятия положительного решения, требования заявления исполняется.

Глава 2. Проектирование системы "обслуживание абонентов"

Rational Rose - мощное CASE-средство для проектирования программных систем любой сложности. Одним из достоинств этого программного продукта будет возможность использования диаграмм на языке UML. Можно сказать, что Rational Rose является графическим редактором UML диаграмм.

В распоряжение проектировщика системы Rational Rose предоставляет следующие типы диаграмм, последовательное создание которых позволяет получить полное представление о всей проектируемой системе и об отдельных ее компонентах:

Use case diagram (диаграммы прецедентов);

Deployment diagram (диаграммы топологии);

Statechart diagram (диаграммы состояний);

Activity diagram (диаграммы активности);

Interaction diagram (диаграммы взаимодействия);

Sequence diagram (диаграммы последовательностей действий);

Collaboration diagram (диаграммы сотрудничества);

Class diagram (диаграммы классов);

Component diagram (диаграммы компонент).

Для целей анализа деятельности предприятия все большее распространение получает средство моделирования Rational Rose компании Rational Software.

Rational Rose - мощный инструмент анализа и проектирования объектно-ориентированных программных систем. Он позволяет моделировать системы до написания кода, так что вы можете с самого начала быть уверены в адекватности их архитектуры. С помощью готовой модели недостатки проекта легко обнаружить на стадии, когда их исправление не требует еще значительных затрат. Среда Rational Rose позволяет проектировать варианты использования и их диаграммы для визуализации функциональных возможностей системы.

2.1 Выявление вариантов использования

UML и Rational Rose являются универсальными средствами, которые вполне подходят и для моделирования бизнес-процессов.

Use case diagram (диаграммы прецедентов) - позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций. Каждая такая диаграмма или, как ее обычно называют, каждый Use case - это описание сценария поведения, которому следуют действующие лица (Actors). Данный тип диаграмм используется при описании бизнес процессов автоматизируемой предметной области, определении требований к будущей системе. Отражает объекты как системы, так и предметной области и задачи, ими выполняемые.

2.1.1 Выделение субъектов (актеров) и прецедентов (видов деятельности)

Исходя из поиска ответов на следующие вопросы:

Кто взаимодействует с системой или использует систему?

Кто передает или принимает информацию в/из системы?

Кто является внешним по отношению к системе?

Я выявил следующих субъектов.

Рис.3 Субъекты системы "Обслуживание абонентов"

Прецедент представляет собой целостный набор функций, имеющих определенную ценность для субъекта. Прецеденты можно вывести в результате определения задач для субъекта. Для этого следует задаться вопросом: "Каковы обязанности субъекта по отношению к системе и чего он ожидает от системы?" Каждый вариант использования (прецедент) определяет набор действий, совершаемых системой при диалоге с актером. При этом ничего не говорится о том, как конкретно будет реализовано взаимодействие актеров с системой и собственно выполнение вариантов использования. Прецедент изображается в виде эллипса, внутри или ниже которого помещается имя прецедента.

Рис.4. Прецеденты системы "Обслуживание абонентов"

2.1.2 Документирование прецедентов

Структура документа, описывающего прецеденты, может варьироваться, однако типичное описание должно содержать следующие разделы.

1. Краткое описание.

2. Участвующие субъекты.

3. Предусловия, необходимые для инициирования прецедента.

4. Детализированное описание потока событий, которое включает:

основной поток, который можно разбить для того, чтобы показать подчиненные потоки событий (подчиненные потоки могут быть разделены дальше на еще более мелкие потоки, с целью сделать читаемость документа более удобной);

альтернативные потоки для определения исключительных ситуаций.

5. Постусловия, определяющие состояние системы, по достижении которого прецедент завершается.

Приведу описательную документацию выше одного из обозначенных прецедентов.

Документ, содержащий описания прецедента, развивается по ходу разработки. На ранней стадии определения требований составляется только краткое описание. Остальные части документа создаются постепенно и итеративно. Полный документ возникает в конце этапа спецификации требований.

Прецедент	Заключение договора
Краткое описание	Данный прецедент необходим для регистрации нового абонента в сети.
Субъект	Оператор, клиент
Предусловия	Оператору необходимо ознакомить с имеющимися операторами связи и выдать форму анкеты потенциальному абоненту
Основной поток	После выбора клиентом соответствующего оператора, он заполняет форму, после чего оператор проверяет правильность заполнение формы на бумажном носителе и вводит данные в систему следующим действием "Выбор оператора связи - Договор об оказании

	услуг связи". После чего в системе "Обслуживание абонентов" открывается форма по заключению абонента сети в системе. При этом сначала система спрашивает, кто будет регистрироваться: Физическое лицо или Юридическое, и только после этого выводится соответствующая регистрационная форма. Оператор вводит информацию об организации-клиенте, о контактном лице юридического лица, а также вводит номера счетов организации. Далее договор сохраняется. Производится сеанс связи с Сервером в процессе которого эти данные передаются на сервер.
Альтернативный поток	В случае, если пользователь не ввел все поля, система выдает сообщение "Введите все поля", дает возможность пройти процесс регистрации снова при ошибке. Также оператор и имеет возможность отказа от регистрации абонента путем выбора соответствующей команды (Отмена или аналогичной).
Постусловия	После успешного завершения прецедента, клиент внесен в базу данных Абоненты сети на сервере.

Прецедент	Замена абонентского номера
Краткое описание	Данный прецедент необходим для удовлетворения потребности абонента, а именно, замены его номера.
Субъект	Оператор, клиент
Предусловия	Оператору необходимо ознакомиться с заявлением клиента.
Основной поток	После проверки твердой копии заявления оператор в БД системы находит данного абонента, вызывает функцию "замена номера", после чего появляется соответствующая форма, которую необходимо заполнить. При нажатии кнопка ОК (или аналогичной) система автоматически меняет в БД номер клиента, оставляя при этом остальные данные неизменными. Далее изменения сохраняются. Производится сеанс связи с Сервером в процессе которого эти данные передаются на сервер.
Альтернативный поток	В случае, если пользователь не ввел все поля либо, система выдает сообщение об ошибке.
Постусловия	После успешного завершения прецедента, внесены изменения в базу данных.

2.1.3 Диаграмма прецедентов

Этот вид диаграмм позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций.

Каждая такая диаграмма или, как ее обычно называют, каждый Use case - это описание сценария поведения, которому следуют действующие лица (Actors).

Диаграмма прецедентов приписывает прецеденты к субъектам. Она также позволяет пользователю установить отношения между прецедентами, конечно, если такие отношения существуют.

Диаграмма прецедентов - это наглядное представление субъектов и прецедентов вместе с любыми дополнительными определениями и спецификациями. На данном виде диаграмм отображаются основные функции, которые выполняет система, лица, оказывающие влияния на систему - внешние сущности, а также связи между ними. Диаграмма прецедентов представляет собой не просто некую схему, а является полностью документированной моделью предполагаемого поведения системы. Достоинства модели вариантов использования заключаются в том, что она:

определяет пользователей и границы системы;

определяет системный интерфейс;

удобна для общения пользователей с разработчиками;

используется для написания тестов;

является основой для написания пользовательской документации;

хорошо вписывается в любые методы проектирования (как объектно-ориентированные, так и структурные).

Варианты использования и субъекты, выделенные для данной модели, можно представить в виде следующей диаграммы прецедентов:

Рис.5. Диаграмма прецедентов системы "Обслуживание абонентов"

Диаграмма прецедентов представляет собой не просто некую схему, а является полностью документированной моделью предполагаемого поведения системы. Это вид диаграмм особенно важен при организации и моделирования поведения системы. На них представлены прецеденты и актеры (частный случай классов), а также отношения между ними. Актер - это роль, которую пользователь играет по отношению к системе.

Между субъектами и вариантами использования могут быть различные виды взаимодействия. Так, из построенной диаграммы видно, что Оператор, инициирует различные варианты использования: Рассмотрение заявления, Замена абонентского номера, Детализация счета, Замена SIM-карты, Блокировка номера и так далее. Клиент также может инициировать варианты использования, например, Заполнение заявления, Составление анкеты, Выбор оператора связи и т.д. Остановлюсь подробнее на некоторых отношениях между вариантами использования.

Рис.6. Диаграмма прецедентов для субъекта Оператор.

Следует прокомментировать некоторые особенно "привлекательные" отношения между вариантами использования.

Так, смысл отношения "***include"*** состоит в том, что *Подключение* включает в себя *Выбор оператора связи*.

Смысл же связи <<***extend***>> в том, что прецедент, например, *Рассмотрение анкеты* "расширяется" вариантом использования *Заключение договора*. Я объясняю это тем, что Заключить договор можно только после проверки оператором анкеты. *Рассмотрение заявления* "расширяет" прецедент *Блокировка номера, Замена sim-карты, Детализация счета, Замена абонентского номера*. Таким образом, связь <<***extend***>> говорит о выполнении того или иного прецедента в зависимости от определенных условий.

Рис.7. Диаграмма прецедентов для субъекта Клиент.

Подключение абонента включает в себя Выбор оператора связи.

Составление анкеты "расширяется" вариантом использования Заключение договора.

2.2 Выявление классов - сущностей

Параллельно с моделированием вариантов использования выполняется выявление так называемых классов-сущностей, их атрибутов и взаимосвязей между ними, что представляется в виде **диаграммы классов (Class diagram),** использующейся для моделирования статического видения системы с точки зрения проектирования, т.е. для построения логической модели разрабатываемой системы. Она не содержит информации о временных аспектах функционирования системы. Каждый класс рассматривается в разрезе нескольких функциональных требований.

***Класс (class*)** в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов.

Класс-сущность - класс, определяющий типы объектов системы и различного рода связи, существующие между ними. Классы-сущности представляют собой элементы, используемые системой постоянно. Они хранят в определенный момент времени часть БД. Данные в них могут извлекаться, меняться, снова записываться. Анализ требований сводится к выявлению классов-сущностей.

Проводя глубокий анализ предметной области, я выявил следующие классы - сущности: Клиент, Физическое лицо, Юридическое лицо, Договор, Заявление, Оператор, Заявление.

Рис.8. Первоначальный вид классов-сущностей.

Класс "Клиент" необходим для хранения информации о клиентах абонентах) системы. Он включает в себя следующие атрибуты: индекс, страна, город, область, район, улица, дом, корпус, квартира, e-mail, факс, телефон.

Так как клиентом системы может быть как частное лицо, так и организация, мною составлены такие классы как "Физическое лицо" и "Юридическое лицо". Их взаимосвязь с классом "Клиент" я покажу на диаграмме классов.

Класс "договор" является хранилищем договоров об оказаниях услуг связи. Он включает в себя следующие атрибуты: телефонный номер, тарифный план, серийный номер sim - карты, услуги связи, особые условия, срок действия договора, дата заключения.

Услуги связи имеют тип list<integer>, который обозначает тот факт, что клиент может из нескольких видов услуг выбрать нужный из списка.

Телефонный номер имеет тип integer, так как номер состоит только из целых чисел. Классы не существуют, как правило, автономно, а взаимодействуют между собой. Потому далее была построена *диаграмма классов.*

Рис.9. Диаграмма классов

Интересна связь между рассматриваемыми объектами *Физическое лицо* - *Клиент* и *Юридическое лицо* - *Клиент*. Это отношение, называется *Обобщение*. Оно представляет собой видовое отношение между более общим классом (суперкласс или родительский класс) и более специфическим видом класса (подкласс или дочерний класс). Подкласс является видом суперкласса. Там, где допустимо использование суперкласса, может использоваться и объект подкласса.

Обобщение делает невозможным переопределение уже заданных свойств. Атрибуты и операции, уже определенные для суперкласса, могут повторно использоваться в подклассе. Говорят, что подкласс наследует (inherit) атрибуты и методы его родительского класса. Обобщение способствует пошаговой спецификации, использованию общих свойств разными классами и лучшей локализации изменений. Обобщение изображается в виде незаполненного треугольника на конце линии отношения, присоединенной к родительскому классу. На диаграмме классов *Клиент* является суперклассом, а *Физическое лицо* и *Юридическое лицо* - подклассами. Классы *Физическое лицо* и *Юридическое лицо* наследуют все атрибуты класса *Клиент*.

Центральным классом рассматриваемой модели стал класс *Договор*. Факт того, что документы создаются Клиентами, а вносятся в систему Операторами показан на диаграмме существованием связи между объектами *Клиент* - Договор и Договор-Оператор. Кратность этой ассоциации вполне объяснима: Клиент может создать один документ или их множество (кратность в этой позиции - [1. n]), и он обязательно должен быть создан (и впоследствии введен оператором в систему), на что и указывает кратность ассоциации [1. n].

Класс Договор служит для хранения всех договоров об оказании услуг связи, по структуре абсолютно идентичных. Поэтому различие между ними устанавливается посредством атрибута тип платежного документа.

Клиент может в силу определенных обстоятельств составить заявление, но также заявление может быть и не составлено, на что указывает кратность [0. .1].

Итак, формирование диаграммы классов-сущностей окончено. Мною были определены типы объектов, определяющих будущую модель базы данных, а также связи, существующие между ними. Однако построенную диаграмму классов нельзя назвать полной статической моделью системы в силу отсутствия многих важных элементов, таких как управляющие и интерфейсные классы.

2.3 Моделирование видов деятельности

Модели видов деятельности (*Activity Diagram*) строятся для описания общей последовательности действий для нескольких объектов и вариантов использования. На диаграммах этого типа представляются переходы потока управления от одной деятельности к другой внутри системы. Этот вид диаграмм относится к динамическим представлениям системы, и является наиболее полезным при моделировании ее функционирования, так как отражает передачу потока управления между объектами.

Основным элементом диаграммы видов деятельности является деятельность. Интерпретация этого термина зависит от той точки зрения, с которой строится данная диаграмма. На концептуальной диаграмме деятельность - это некоторая задача, которую необходимо выполнить вручную или автоматизированным способом. На диаграмме, построенной в аспекте спецификации или реализации, деятельность представляет собой некоторый метод над классом.

Диаграмма деятельностей предоставляет свободу выбора порядка выполнения. Другими словами, она только устанавливает основные правила последовательности, которым необходимо следовать. Такая возможность важна при моделировании бизнес-процессов. Среди бизнес-процессов нередко встречаются такие, которые не обязаны выполняться

последовательно. В таких ситуациях данный метод хорошо работает, так как он позволяет реализовывать процессы параллельно.

Диаграммы видов деятельности являются также полезными при параллельном программировании, поскольку можно графически изобразить все ветви и определить, когда их необходимо синхронизировать.

Если при описании поведения системы имеются параллельные деятельности, то их необходимо синхронизировать. Простая линейка синхронизации показывает, что ее выходная деятельность активизируется только тогда, когда выполнены обе входные деятельности.

Диаграммы видов деятельности можно применять для описания потоков событий в вариантах использования. С помощью текстового описания можно достаточно подробно рассказать о потоке событий, но в сложных и запутанных потоках с множеством альтернативных ветвей будет трудно понять логику событий. Диаграммы видов деятельности предоставляют ту же информацию, что и текстовое описание потока событий, но в наглядной графической форме.

Рассмотрим теперь модели видов деятельности, построенные для проектируемой системы.

Составление анкеты:

Рис.10. Диаграмма видов деятельности для прецедента Составление анкеты

Клиент получает форму для заполнения от оператора, после чего ему

необходимо заполнить все поля данной формы. Далее оператор проверяет анкету, в случае наличия ошибок клиенту выдается новая форма. При одобрении - заключается договор об оказании услуг связи.

Рассмотрение анкеты:

Рис.11. Диаграмма видов деятельности для прецедента Рассмотрение анкеты

Оператор получает от клиента заполненную анкету, проверяет все поля. При наличии ошибок отказывает в заключении договора. В противном случае заключает договор.

Замена абонентского номера:

Рис.12. Диаграмма видов деятельности для прецедента Замена абонентского номера

Оператор получает от клиента заявление на замену абонентского номера, проверяет на наличие ошибок в реквизитах. Если они есть, то выдается новая форма заявления, иначе, рассматривается причина. В результате рассмотрения причины заявление могут удовлетворить, либо отклонить.

Детализация счета:

Рис.13. Диаграмма видов деятельности для прецедента Детализация счета

Оператор получает от клиента заявление на получение детализации счета, проверяет соответствие личности и паспорта. Если паспорт не принадлежит будущему абоненту, ему отказывают. Далее идет проверка на наличие ошибок в заявлении. При их отсутствии выдается детализация счета за необходимый период. При наличии ошибок выдается новая форма заявления.

Замена SIM-карты:

Рис.14. Диаграмма видов деятельности для прецедента Замена SIM-карты

Оператор получает от клиента заявление на Замену sim-карты, проверяет соответствие личности и паспорта. Если паспорт не принадлежит будущему абоненту, ему отказывают. Далее идет проверка на наличие ошибок в заявлении. При их отсутствии рассматривается причина, если она удовлетворительная, то карты заменят. При наличии ошибок выдается новая форма заявления.

Выбор оператора связи:

Рис.15. Диаграмма видов деятельности для прецедента Выбор оператора связи

Выбор оператора связи после ознакомления со всеми возможными вариантами, изучение тарифов. Далее заключается договор об оказании услуг связи и абонент может подключиться к сети.

2.4 Моделирование взаимодействий

Взаимодействие объектов в системе происходит посредством приема и передачи сообщений объектами-клиентами и обработки этих сообщений объектами-серверами. Чтобы отразить последовательность передачи сообщений между объектами, для каждого прецедента использования может быть построена модель динамического взаимодействия объектов, которая представляется в одной из двух форм:

в форме диаграммы последовательностей (sequence diagram), на которой основное внимание уделяется временной упорядоченности событий. На них изображают множество объектов и посланные или принятые ими сообщения. Объекты, как правило, представляют собой экземпляры классов.

в форме диаграммы кооперации (collaboration diagram), которая отражает структурную организацию объектов, принимающих или отправляющих сообщения. На диаграмме кооперации показано множество объектов, связи между ними и сообщения, которые они посылают или получают.

Диаграммы последовательностей и кооперации относятся *к динамическому виду* системы. Они *изоморфны*, то есть их можно преобразовывать друг в друга без потери информации. Поэтому мною в

работе будет рассмотрена лишь диаграмма последовательности, из которой при желании можно получить и кооперацию объектов.

Для диаграммы последовательности ключевым моментом является динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения. Одно представлено слева направо в виде ***линии жизни*** *(lifeline) (*период времени существования) отдельного объекта, участвующего во взаимодействии, а второе - вертикальной временной осью, направленной сверху вниз. Взаимодействие объектов реализуется посредством сообщений, посылаемые одними объектами другим. Сообщения появляются в том порядке, в котором они показаны - сверху вниз.

Заключение договора:

Рис.16. Диаграмма последовательности Заключение договора

Для Заключения договора клиенту необходимо передать подписанный договор оператору, который в последствии открывает главное окно. Далее он должен выбрать регистрацию. Вводятся сведения в пограничный класс Окно регистрации окно составления договора, заполняет появившуюся форму. Система управления проверяет введенные данные и сохраняет договор. Затем в контейнерном классе Клиент сохраняется информация о новом клиенте.

Ниже приведена диаграмма кооперации для рассмотренного варианта использования

Заключение договора:

Рис.17. Диаграмма кооперации Заключение договора

Детализация счета:

Рис.18. Диаграмма последовательности Детализация счета

Оператор открывает главное окно системы, затем окно документа, а именно окно детализации счета. Далее формирует отчет путем передачи сообщения управляющему классу Система управления критерии времени. В окне детализации отображается перечень всех звонков, удовлетворяющих требованию клиента. Документ, отобразившийся в окне, оператор может распечатать.

Ниже приведена диаграмма кооперации для рассмотренного варианта использования.

Блокировка номера:

Рис. 19. Диаграмма последовательности Блокировка номера

Оператор открывает главное окно системы, затем окно документа, а именно окно блокировки. Далее путем передачи сообщения управляющему классу Система управления, блокирует номер. Система управления выполняет заданный запрос, сохраняет свои действия и информирует об этом.

Ниже приведена диаграмма кооперации для рассмотренного варианта использования.

Блокировка номера:

Рис. 20. Диаграмма кооперации Блокировка номера

2.5 Моделирование состояний

Следующим этапом разработки является моделирование состояний, необходимая информация для которого содержится в модели классов-сущностей и модели взаимодействий. Модель классов представляет классы, которые подлежат анализу на предмет выявления нетривиального поведения, а модель взаимодействий помогает определить методы, которые могут влиять на изменение состояний классов.

Состояние можно рассматривать как отдельную ситуацию, в течение которой имеет место выполнение некоторого условия. Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта.

Следует заметить, что не каждый атрибут класса может характеризовать его состояние. Как правило, имеют значение только такие свойства элементов системы, которые отражают динамический или функциональный аспект ее поведения.

Диаграммы состояний (*Statechart Diagram*) определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий. В большинстве объектно-ориентированных методов диаграммы состояний строятся для единственного класса и отражают динамику поведения единственного объекта.

Не следует строить диаграммы состояний для каждого класса в системе; их стоит использовать только для тех классов, поведение которых действительно интересует, и построение диаграмм состояний помогает лучше его понять.

На основании анализа проектируемой системы и требований к ней, можно сказать, что интересным разработчику представляется поведение класса Договор.

Рассмотрю диаграмму состояний для этого класса.

Рис.21. Диаграмма состояний для класса "Договор"

Перед созданием Договора он приобретает состояние *Новый.* При подписании договора обеими сторонами переходит в состояние *Подписан*. *Действителен* он будет до срока действия, после чего он может перейти в состояние *Продлен* либо *Просрочен,* Расторгнут и, соответственно, перейдет в состояние *Удален*.

Рис.22. Диаграмма состояний для класса "Заявление"

Перед созданием и в процессе создания заявления, он приобретает состояние *Новый*. После подписи клиентом переходит в состояние Заполненное. При передаче его оператору, заявление переходит с состояние На рассмотрении. Если на стадии рассмотрения обнаружены ошибки, то заявление переходит в состояние Отвергнут, иначе - в состояние На обработке. При удовлетворении заявления он Исполнен, в противном случае - отвергнут.

2.6 Проектирование статической структуры

Статическая структура информационной системы представляет собой конечную диаграмму классов, на которой представлено взаимодействие сущностей, управляющих и пограничных классов, выявленных на начальном этапе проектирования и в процессе моделирования взаимодействий. Модель взаимодействий служит источником информации не только о том, какие классы, помимо сущностей, выявленных в начале разработки, должны существовать в системе, но и о том, как они взаимодействуют и связаны друг с другом, а также какими методами они обладают.

На диаграммах ниже статическая структура представлена в двух частях. Первая иллюстрирует взаимодействие управляющего класса с сущностями, а вторая - взаимодействие управляющего класса с пограничными.

Рис.23. Статическая структура ИС (часть 1)

Рис.24. Статическая структура ИС (часть 2)

Не все модели последовательностей и кооперации были построены с учетом принципа трехслойного взаимодействия (что, возможно, является недостатком данной разработки), поэтому на конечной диаграмме классов должны присутствовать связи между некоторыми пограничными классами и сущностями. Однако ввиду того, что при отображении всех этих связей на одной диаграмме ее практически невозможно будет читать, она в работе не приводится

2.7 Проектирование пользовательского интерфейса

После завершения предыдущих этапов разработки имею всю необходимую информацию для того, чтобы приступить к следующему не менее важному - к проектированию пользовательского интерфейса.

Пользователи программ, как правило, не разделяют функциональность и пользовательский интерфейс. Для пользователей именно пользовательский интерфейс является программой. Для них, если интерфейс хороший, стало быть, и сама программа хороша и удобна.

Пользовательский интерфейс часто понимают только как внешний вид системы. В действительности пользовательский интерфейс включает в себя все аспекты дизайна, которые оказывают влияние на взаимодействие пользователя и системы. Это не только экран, который видит пользователь. Пользовательский интерфейс состоит из множества составляющих, таких как набор задач пользователя, которые он решает при помощи системы, элементы управления системой, навигация между блоками системы, визуальный (и не только) дизайн экранов программы. При проектировании пользовательского интерфейса я опиралась на модель взаимодействий, которая предоставляет информацию о том, какие окна (в данном случае формы) отображаются в ответ на то или иное действие пользователя и какие элементы в них должны содержаться. Кроме того, разработку пользовательского интерфейса я

постарался провести с учетом принципов проектирования интерфейса. При построении в интерфейсе были использованы термины и понятия, доступные и понятные будущим пользователям системы. Я сделать так, чтобы разные, но однотипные операции проводились аналогичным способом. Кроме того, интерфейс предоставляет необходимую информацию в случае возникновения каких-либо ошибок. Вся информация о сообщениях об ошибках также представлена на моделях взаимодействий. Ввиду большого объема работы мною была спроектирована лишь часть пользовательского многодокументного интерфейса. Далее результат выполнения этого этапа проектирования.

Рис.25. Модель интерфейса для заключения договора

Так как интерфейс программы многодокументный, то все дочерние окна

открываются и сворачиваются в родительском окне в свободной площади. Дочерние окна могут не открываться, выноситься вне родительского.

Меню - в данном компоненте содержатся все основные команды доступные пользователю при работе с системой.

Панель инструментов - на этом графическом элементе размещены кнопки, ассоциированные с наиболее часто применяемыми командами.

2.8 Диаграмма компонентов

Для создания конкретной физической системы необходимо реализовать все элементы логического представления в конкретные материальные сущности. Для описания таких реальных сущностей предназначено физическое представление модели. Базовыми элементами физического представления системы в нотации UML являются компоненты, которые представляют собой физически существующие части системы, которые обеспечивают реализацию классов и отношений, а также функционального поведения моделируемой программной системы.

На рисунке 26 показан результат попытки, определить состав компонентов системы и представить их в виде диаграммы компонентов.

Рис.26. Диаграмма компонентов

2.9 Проектирование архитектуры приложения

Технология клиент-сервер по праву считается одним из "китов", на которых держится современный мир компьютерных сетей.

Для проектирования архитектуры данного приложения я безусловно использую технологию клиент-сервер. Что же касается архитектуры, двухуровневая или трех, то здесь есть над чем порассуждать. Итак, термин "клиент-сервер" означает такую архитектуру программного комплекса, в которой его функциональные части взаимодействуют по схеме "запрос-ответ". Если рассмотреть две взаимодействующие части этого комплекса, то одна из них (клиент) выполняет активную функцию, т.е. инициирует запросы, а другая (сервер) пассивно на них отвечает. По мере развития системы роли могут меняться, например некоторый программный блок будет одновременно выполнять функции сервера по отношению к одному блоку и клиента по отношению к другому.

Замечу, что любая информационная система должна иметь минимум три основные функциональные части - модули хранения данных, их обработки и интерфейса с пользователем, в моем случае с оператором. Каждая из этих частей может быть реализована независимо от двух других. Например, не изменяя программ, используемых для хранения и обработки данных, можно изменить интерфейс с оператором таким образом, что одни и те же данные будут отображаться в виде таблиц, графиков или гистограмм. Не меняя программ представления данных и их хранения, можно изменить программы обработки, например, изменив алгоритм полнотекстового поиска. И наконец, не меняя программ представления и обработки данных, можно изменить программное обеспечение для хранения данных, перейдя, например, на другую файловую систему.

В классической архитектуре клиент-сервер приходится распределять три основные части приложения по двум физическим модулям. Обычно ПО

хранения данных располагается на сервере (например, сервере базы данных), интерфейс с пользователем - на стороне клиента, а вот обработку данных приходится распределять между клиентской и серверной частями. В этом-то и заключается основной недостаток двухуровневой архитектуры, из которого следуют несколько неприятных особенностей, сильно усложняющих разработку клиент-серверных систем.

При разбиении алгоритмов обработки данных необходимо синхронизировать поведение обеих частей системы. Все разработчики должны иметь полную информацию о последних изменениях, внесенных в систему, и понимать эти изменения. Это создает большие сложности при разработке клиент-серверных систем, их установке и сопровождении, поскольку необходимо тратить значительные усилия на координацию действий разных групп специалистов. В действиях разработчиков часто возникают противоречия, а это тормозит развитие системы и вынуждает изменять уже готовые и проверенные элементы.

Чтобы избежать несогласованности различных элементов архитектуры, пытаются выполнять обработку данных на одной из двух физических частей - либо на стороне клиента ("толстый" клиент), либо на сервере ("тонкий" клиент). Каждый подход имеет свои недостатки. В первом случае неоправданно перегружается сеть, поскольку по ней передаются необработанные, а значит, избыточные данные. Кроме того, усложняется поддержка системы и ее изменение, так как замена алгоритма вычислений или исправление ошибки требует одновременной полной замены всех интерфейсных программ, а иначе могут возникнуть ошибки или несогласованность данных. Если же вся обработка информации выполняется на сервере (когда такое вообще возможно), то возникает проблема описания встроенных процедур и их отладки. Дело в том, что язык описания встроенных процедур обычно является декларативным и, следовательно, в принципе не допускает пошаговой отладки. Кроме того, систему с обработкой информации на сервере абсолютно невозможно перенести на другую платформу, что является серьезным недостатком.

Большинство современных средств быстрой разработки приложений (RAD), которые работают с различными базами данных, реализует первую стратегию, т.е. "толстый" клиент обеспечивает интерфейс с сервером базы данных через встроенный SQL. Такой вариант реализации системы с "толстым" клиентом, кроме перечисленных выше недостатков, обычно обеспечивает недопустимо низкий уровень безопасности. Например, в банковских системах приходится всем операционистам давать права на запись в основную таблицу учетной системы. Кроме того, данную систему почти невозможно перевести на Web-технологию, так как для доступа к серверу базы данных используется специализированное клиентское ПО.

Итак, рассмотренные выше модели имеют следующие недостатки.

1. "Толстый" клиент:

сложность администрирования;

усложняется обновление ПО, поскольку его замену нужно производить одновременно по всей системе;

усложняется распределение полномочий, так как разграничение доступа происходит не по действиям, а по таблицам;

перегружается сеть вследствие передачи по ней необработанных данных;

слабая защита данных, поскольку сложно правильно распределить полномочия.

2. "Толстый" сервер:

усложняется реализация, так как языки типа PL/SQL не приспособлены для разработки подобного ПО и нет хороших средств отладки;

производительность программ, написанных на языках типа PL/SQL, значительно ниже, чем созданных на других языках, что имеет важное значение для сложных систем;

программы, написанные на СУБД-языках, обычно работают недостаточно надежно; ошибка в них может привести к выходу из строя всего сервера баз данных;

получившиеся таким образом программы полностью непереносимы на другие системы и платформы.

Для решения перечисленных проблем использую трехуровневую архитектур клиент-сервер.

В трехуровневой архитектуре "тонкий" клиент не перегружен функциями обработки данных, а выполняет свою основную роль системы представления информации, поступающей с сервера приложений. Трехуровневая архитектура клиент-сервер позволяет более точно назначать полномочия операторов, так как они получают права доступа не к самой базе данных, а к определенным функциям сервера приложений. Это повышает защищенность системы (по сравнению с обычно архитектурой) не только от умышленного нападения, но и от ошибочных действий персонала.

Для примера рассмотрю систему, различные части которой работают на нескольких удаленных друг от друга серверах. Допустим, что от разработчика поступила новая версия системы, для установки которой в двухуровневой архитектуре необходимо одновременно поменять все системные модули. Если же этого не сделать, то взаимодействие старых клиентов с новыми серверами может привести к непредсказуемым последствиям, так как разработчики обычно не рассчитывают на такое использование системы. В трехуровневой архитектуре ситуация упрощается. Дело в том, что поменяв сервер приложений и сервер хранения данных (это легко сделать одновременно, так как оба они обычно находятся рядом), мы сразу меняем набор доступных сервисов. Таким образом, вероятность ошибки из-за несоответствия версий серверной и клиентской частей резко сокращается. Если в новой версии какой-либо сервис исчез, то элементы интерфейса, обслуживавшие его в старой системе, просто не будут работать. Следует отметить и тот факт, что в трехуровневой системе по каналу связи между сервером приложений и базой данных передается достаточно много информации. Однако это не замедляет вычислений, так как для связи указанных элементов можно использовать более скоростные линии. Это потребует минимальных затрат, поскольку оба сервера обычно находятся в

одном помещении. Таким образом, увеличивается суммарная производительность системы - над одной задачей теперь работают два различных сервера, а связь между ними можно осуществлять по наиболее скоростным линиям с минимальными затратами средств.

Рис.27. Диаграмма развертывания

Заключение

В результате проделанной мною работы была спроектирована имитационная модель информационной системы "Центр обслуживания абонентов".

В процессе работы были созданы модели вариантов использования, классов-сущностей, видов деятельности, взаимодействий, состояний, статической структуры, пользовательского интерфейса, компонентов, и архитектуры приложения.

Хочу отметить, что разработанный проект далеко не является полным и готовым к реализации. В процессе разработки были рассмотрены и смоделированы основные функции системы. Но помимо рассмотренных, система должна поддерживать еще и другие функции. Следует сказать, что плохо проработана диаграмма компонентов, однако это можно объяснить тем, что для определения состава компонентов, из которых должна состоять система, необходим очень большой объем знаний, на приобретение которого практически не было времени.

Дипломная работа был выполнена с использованием case-средств BPWin 4.0 и IBM Rational Rose 2003.

Глоссарий

CASE (**C**omputer-**A**ided **S**oftware/**S**ystem **E**ngineering) - средства автоматизации методологий структурного системного анализа и проектирования

СУБД - система управления базами данных

БД - база данных

ТфОП - телефонной сети общего пользования

SADT (Structured Analysis and Design Technique) - технология структурного анализа и проектирования и ее подмножество стандарт IDEF0

DFD (Data Flow Diagrams) - диаграммы потоков данных

ERD (Entity-Relationship Diagrams) - диаграммы "сущность-связь"

STD (State Transition Diagrams) - диаграммы переходов состояний.

ПО - программное обеспечение

Библиография

1. Д.Н. Столбовский. Лекции по проектированию экономических информационных систем.
2. Леоненков. Самоучитель UML.
3. Е.Б. Золотухина, Р.В. Алфимов (с) Interface Ltd., 2001.
4. Доклад компании Yphise "UML-моделирование информационных систем и проектов" ("UML Modeling of Projects and Information Systems") / www.interface.ru/
5. Липаев В.В. "Качество программного обеспечения". - М.: "Финансы и статистика", 1993.
6. Боэм Б.У. "Инженерное проектирование программного обеспечения". - М.: "Радио и связь", 1985.
7. "Технико-экономическое обоснование дипломных проектов" под редакцией проф. В.К. Беклешова. - М.: "Высшая школа", 1991.
8. Костров А.В. "Основы информационного менеджмента": Учебное пособие. - М.: Финансы и статистика, 2001.
9. Классификационный справочник должностей служащих. - М.: ИНФРА-М, 2001.