|  |  |
| --- | --- |
| Описание: Описание: Описание: http://almetpt.ru/img/emblema.png | Министерство образования и науки Республики Татарстан  Государственное автономное профессиональное образовательное учреждение  **«Альметьевский политехнический техникум»** (ГАПОУ «АПТ») |

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ**

**ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ**

МДК.02.01. Технология разработки программного обеспечения

по специальности 09.02.07 Информационные системы и программирование

2020 г.

|  |  |
| --- | --- |
| **ОДОБРЕНО** | **УТВЕРЖДАЮ** |
| Цикловой комиссией | Зам. директора по УР |
| информационно-коммуникационных  дисциплин | \_\_\_\_\_\_\_ Р.М.Бородина |
| Председатель | 05 сентября 2020г. |
| \_\_\_\_\_\_\_\_\_Ю.А.Рамазанова |  |
| \_\_\_\_ \_\_\_\_\_\_\_\_\_\_2020г. |  |
|  |  |

Составитель:

Филимонова А.Ю.. преподаватель информационных технологий ГАПОУ «Альметьевский политехнический техникум»;

Эксперты:

Внутренняя экспертиза

Техническая экспертиза: Захарова И.М., методист ГАПОУ «Альметьевский политехнический техникум»;

Содержательная экспертиза: Рамазанова Ю.А., председатель ЦК ГАПОУ «Альметьевский политехнический техникум».

Внешняя рецензия: Куприянов Д.А., ведущий инженер ООО «Процессинговый Центр», Центр АСУТП.

**Содержание**

[Лабораторная работа №1. Анализ предметной области 4](#_Toc69194996)

[Лабораторная работа №2-3. Разработка и оформление технического задания. 11](#_Toc69194997)

[Лабораторная работа №4. Разработка архитектуры программного средства. 20](#_Toc69194998)

[Лабораторная работа №5. Изучение работы в системе контроля версий 27](#_Toc69194999)

[Лабораторная работа №6. Построение диаграмм Вариантов использования и Последовательности 36](#_Toc69195000)

[Лабораторная работа №7. Построение диаграмм Кооперации и Развертывания 58](#_Toc69195001)

[Лабораторная работа №8. Построение диаграмм Деятельности, Состояния и Классов. 77](#_Toc69195002)

[Лабораторная работа №9. Построение диаграммы Компонентов 104](#_Toc69195003)

[Лабораторная работа №10. Построение диаграммы Потоков данных 113](#_Toc69195004)

[Лабораторная работа №11. Разработка тестового сценария 116](#_Toc69195005)

[Лабораторная работа №12. Оценка необходимого количества тестов 118](#_Toc69195006)

[Лабораторная работа №13. Разработка тестовых пакетов. 122](#_Toc69195007)

[Лабораторная работа №14. Оценка программных средств с помощью метрик. 129](#_Toc69195008)

[Лабораторная работа №15. Инспекция программного кода на предмет соответствия стандартам кодирования. 150](#_Toc69195009)

**Лабораторная работа №1****. Анализ предметной области**

**Цель работы**: провести анализ и оформить результаты обследования деятельности предприятия; осуществлять постановку задач по обработке информации; проводить анализ предметной области.

**Ход работы**

В процессе выполнения лабораторной работы проводится анализ и оформление результатов обследования деятельности гипотетического предприятия "МЕД", и на его основе разрабатываются документы, необходимые для настройки типовой ИС.

По итогам проведения обследования обычно формируется отчет об обследовании.

Обследование начинается со сбора предварительной информации о компании. Итогом являются следующие данные:

* краткая информация о компании (профиль клиента).
* цели проекта.
* подразделения и пользователи системы.

В процессе выполнения лабораторной работы проводится анализ и оформление результатов обследования деятельности гипотетического предприятия "МЕД", и на его основе разрабатываются документы, необходимые для настройки типовой ИС.

По итогам проведения обследования обычно формируется отчет об обследовании.

Обследование начинается со сбора предварительной информации о компании. Итогом являются следующие данные:

* краткая информация о компании (профиль клиента).
* цели проекта.
* подразделения и пользователи системы.

На основе предварительной информации формируется и согласовывается с заказчиком общее представление о проекте:

Видение выполнения проекта и границы проекта - документ, который кратко описывает, в каких подразделениях и в какой функциональности будет внедряться ИС.

Затем выполняется детальное обследование предприятия, результаты которого оформляются в виде отдельного документа - отчета об обследовании.

Отчет об обследовании содержит следующие разделы:

* анализ существующего уровня автоматизации.
* составляется список программного обеспечения, используемого в компании, и приводятся данные об использовании этих пакетов в каждом из подразделений организации.
* общие требования к ИС
* формулируются общие требования к функциональности разрабатываемой системы.
* организационная диаграмма

Организационная диаграмма используется для отражения организационной структуры подразделений предприятия и их зон ответственности.

* описание состава автоматизируемых бизнес-процессов

Все бизнес-процессы компании должны быть перечислены в общем списке и каждый должен иметь свой уникальный номер.

* диаграммы прецедентов.

Для выделения автоматизируемых бизнес-процессов и их основных исполнителей используются диаграммы прецедентов.

* физическая диаграмма

Физическая диаграмма служит для того, чтобы описать взаимодействие организации на верхнем уровне с внешними контрагентами.

* описания бизнес-процессов (книга бизнес-процессов).

Далее в отчет об обследовании включается книга бизнес-процессов, содержащая подробное описание автоматизируемых бизнес-процессов. Модели бизнес-процессов позволяют выделить отдельные операции, выполнение которых должно поддерживаться разрабатываемой ИС.

На последнем этапе осуществляется отображение модели предметной области на функциональность типовой системы - выбираются модули системы для поддержки выделенных операций, определяются особенности их настройки, выявляется необходимость разработки дополнительных программных элементов.

**Пример отчета обследования предметной области.**

Краткая информация о компании "МЕД"

Компания - дистрибьютор "МЕД" закупает медицинские препараты отечественных и зарубежных производителей и реализует их через собственную дистрибьюторскую сеть и сеть аптек. Компания осуществляет доставку товаров как собственным транспортом, так и с помощью услуг сторонних организаций.

Основные бизнес-процессы компании - закупки, складирование запасов, продажи, взаиморасчеты с поставщиками и клиентами.

Уровень конкуренции для компании в последнее время возрос, так как на рынок вышли два новых конкурента, к которым перешла часть клиентов и ряд наиболее квалифицированных сотрудников ЗАО "МЕД". ЗАО "МЕД" имеет два филиала - в Курске и Санкт-Петербурге. Каждый филиал функционирует как самостоятельное юридическое лицо, являясь полностью принадлежащей ЗАО "МЕД" дочерней компанией.

По предварительным планам, Компания намерена открыть также дочернее предприятие для организации производства в непосредственной близости к своим заказчикам.

Адреса и телефоны

Москва, К-123 Центральная улица, д. 20, стр. 7, офис 709

Телефон: (095) 345-6789, факс: (095) 345-9876

Контактные лица

Борис Нефедьев - Генеральный директор

Дмитрий Кононов - Исполнительный директор

Артур Иванченко - Директор по маркетингу

Сотрудники

На момент проведения Диагностики штат компании составляет 110 сотрудников.

Основными целями проекта автоматизации компании "МЕД" являются:

1. Разработка и внедрение комплексной автоматизированной системы поддержки логистических процессов компании.
2. Повышение эффективности работы всех подразделений компании и обеспечение ведения учета в единой информационной системе.

Видение выполнения проекта и границы проекта

В рамках проекта развертывание новой системы предполагается осуществить только в следующих подразделениях ЗАО "МЕД":

* Отдел закупок;
* Отдел приемки;
* Отдел продаж;
* Отдел маркетинга;
* Группа планирования и маркетинга;
* Группа логистики;
* Учетно-операционный отдел;
* Учетный отдел;
* Отдел сертификации (в части учета сертификатов на медикаменты);
* Бухгалтерия (только в части учета закупок, продаж, поступлений и платежей).

Не рассматривается в границах проекта автоматизация учета основных средств, расчета и начисления заработной платы, управления кадрами. Выходит за рамки проекта автоматизация процессов взаимоотношений с клиентами.

Количество рабочих мест пользователей - 50.

Отчет об обследовании

Список программного обеспечения, используемого компанией на момент обследования

1. "1С: Предприятие 7.7" ("Бухгалтерия", "Торговля", "Зарплата", "Кадры", "Касса", "Банк") для работы бухгалтерии.
2. Две собственные разработки на базе конфигуратора "1С" - "Закупки" и "Продажи".
3. Собственная разработка на базе FOXPRO для финансового отдела.
4. Excel для планирования продаж.

Существующий уровень автоматизации представлен в таблице 1.

Таблица 1 – Уровень автоматизации

|  |  |
| --- | --- |
| Количество рабочих станций, всего: | 90 |
| Количество сотрудников отдела IT | 2 |
| Количество ПК, одновременно работающих в сети | 50 |
| Наличие и форма связи с удаленными объектами | Терминальная связь со складом |
| Количество рабочих станций на удаленном объекте | 8 |
| Характеристики компьютеров | От Celeron 600 и выше |
| Операционная система | Windows 98, XP |
| Системы, которые представляется возможным оставить без изменения | "1С: Предприятие 7.7" в модульном составе "Бухгалтерия", "Зарплата", "Кадры", для работы бухгалтерии |

Общие требования к информационной системе

Одно из основных требований компании "МЕД" к будущему решению состоит в том, чтобы оно было построено на фундаменте единой интегрированной системы, а работа всех сотрудников велась в одном информационном пространстве.

Ключевые функциональные требования к информационной системе:

1. Мощные средства защиты данных от несанкционированного доступа. Разграничения доступа к данным в соответствии с должностными обязанностями.
2. Возможность удаленного доступа.
3. Управление запасами. Оперативное получение информации об остатках на складе.
4. Управление закупками. Планирование закупок в разрезе поставщиков.
5. Управление продажами. Контроль лимита задолженности с возможностью блокировки формирования отгрузочных документов.
6. Полный контроль взаиморасчетов с поставщиками и клиентами.
7. Получение управленческих отчетов в необходимых аналитических срезах - как детальных для менеджеров, так и агрегированных, для руководителей подразделений и директоров фирмы.

Организационная диаграмма

Оргструктура предприятия оптовой торговли ЗАО "МЕД" имеет вид, представленный в соответствии с рисунком 1.

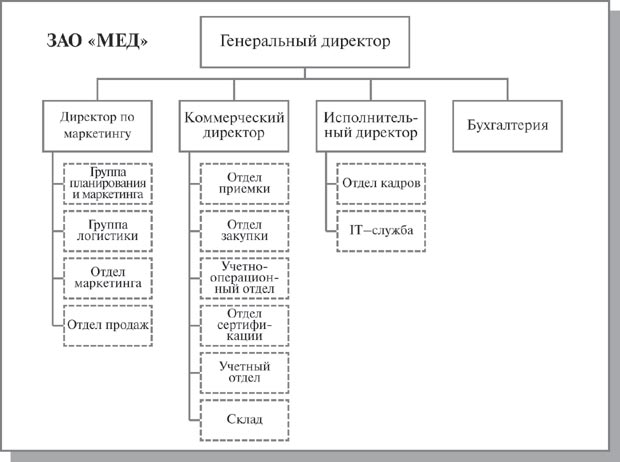


Рисунок 1 – Организационная структура предприятия ЗАО «МЕД»

Описание состава автоматизируемых бизнес-процессов

Бизнес-процессы компании, подлежащие автоматизации, приведены в следующей таблице 6.

Таблица 6 – Бизнес-процессы компании

|  |  |  |
| --- | --- | --- |
| **п.п** | **Код бизнес-процесса** | **Наименование бизнес-процесса** |
| 1. | Закуп-1 | Закупки |
| 2. | Склад-2 | Запасы-Склад |
| 3.  4. | Прод-3  Врасч-4 | Продажи  Взаиморасчеты с поставщиками и клиентами |

Каждый бизнес-процесс имеет свой уникальный номер. Нумерация бизнес-процессов построена по следующему принципу: "префикс-номер", где префикс обозначает группу описываемых бизнес-процессов, а номер - порядковый номер бизнес-процесса в списке.

**Задание для самостоятельного выполнения.**

Задание: провести анализ предметной области по вариантам. Номер варианта соответствует номеру в списке в классном журнале. Список предметных областей приведен в ниже:

1.Компания по добычи нефти и газа

2.Компания по транспортировке нефти и газа

3.Компания по проведению подземного и капитального ремонта скважин

4.Компания по производству и продаже оборудования для добычи нефти и газа

5.Компания по проведению геофизических исследований скважин

6. Компания по производству и продаже оборудования для проведения ремонта скважин

7.Компания по проведению бурения скважин

8. Компания по производству и продаже оборудования для геофизических исследований скважин

9. Компания по разведке месторождений нефти и газа

10.Компания по хранению и складированию нефти и продуктов ее переработки.

11. Компания по исследованию полезных ископаемых и горных пород.

**Требования к отчету**

На протяжении лабораторной работы должен быть получен отчет об обследовании предметной области, содержащий разделы, перечисленные выше.

**Лабораторная работа №2-3. Разработка и оформление технического задания.**

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

**Основные теоретические сведения**

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

*Техническое задание* представляет собой документ, в котором сформулированы основные цели разработки, требования к программному продукту, определены сроки и этапы разработки и регламентирован процесс приемо-сдаточных испытаний. В разработке технического задания участвуют как представители заказчика, так и представители исполнителя. В основе этого документа лежат исходные требования заказчика, анализ передовых достижений техники, результаты выполнения научно-исследовательских работ, предпроектных исследований, научного прогнозирования и т. п.

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

* установить общую цель создания АИС;
* установить общие требования к проектируемой системе;
* разработать и обосновать требования, предъявляемые к информационному, математическому, программному, техническому и технологическому обеспечению;
* определить состав подсистем и функциональных задач;
* разработать и обосновать требования, предъявляемые к подсистемам; − определить этапы создания системы и сроки их выполнения;
* провести предварительный расчет затрат на создание системы и определить уровень экономической эффективности ее внедрения;
* определить состав исполнителей.

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

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

* Титульной;
* Информационной;
* Основной.

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

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

Содержание включает перечень записей о структурных элементах основной части документа.

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

Основная часть технического задания должна содержать следующие разделы:

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

* введение;
* основания для разработки;
* назначение разработки;
* требования к программному продукту;
* требования к программной документации;
* технико-экономические показатели; − стадии и этапы разработки;
* порядок контроля и приемки;

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

**Состав и содержание технического задания (ГОСТ 34.602—89)**

|  |  |
| --- | --- |
| Раздел | Содержание |
| Общие сведения | Полное наименование системы и ее условное обозначение. Шифр темы или шифр (номер) договора. Наименование предприятий разработчика и заказчика системы, их реквизиты. Перечень документов, на основании которых создается ИС. Плановые сроки начала и окончания работ. Сведения об источниках и порядке финансирования работ. Порядок оформления и предъявления заказчику результатов работ по созданию системы, ее частей и отдельных средств |
| Назначение и цепи создания (развития) системы | Вид автоматизируемой деятельности. Перечень объектов, на которых предполагается использование системы. Наименования и требуемые значения технических, технологических, производственно-экономических и др. показателей объекта, которые должны быть достигнуты при внедрении 1/1С |
| Характеристика объектов автоматизации | Краткие сведения об объекте автоматизации. Сведения об условиях эксплуатации и характеристиках окружающей среды |
| Требования к системе | Требования к системе в целом:  − требования к структуре и функционированию системы (перечень подсистем, уровни иерархии, степень централизации, способы информационного обмена, режимы функционирования, взаимодействие со смежными системами, перспективы развития системы);  − требования к персоналу (численность пользователей, квалификация, режим работы, порядок подготовки);  − показатели назначения (степень приспособляемости системы к изменениям процессов управления и значений параметров)  − требования к надежности, безопасности, эргономике, транспортабельности, эксплуатации, техническому обслуживанию и ремонту, защите и сохранности информации, защите от внешних воздействий, к патентной чистоте, по стандартизации и унификации.  Требования к функциям (по подсистемам):  − перечень подлежащих автоматизации задач;  − временной регламент реализации каждой функции;  − требования к качеству реализации каждой функции, к форме представления выходной информации, характеристики точности, достоверности выдачи результатов;  − перечень и критерии отказов.  Требования к видам обеспечения:  − математическому (состав и область применения математических моделей и методов, типовых и разрабатываемых алгоритмов);  − информационному (состав, структура и организация данных, обмен данными между компонентами системы, информационная совместимость со смежными системами, используемые классификаторы, СУБД, контроль данных и ведение информационных массивов, процедуры придания юридической силы выходным документам);  − лингвистическому (языки программирования, языки взаимодействия пользователей с системой, системы кодирования, языки ввода-вывода);  − программному (независимость программных средств от платформы, качество программных средств и способы его контроля, использование фондов алгоритмов и программ);  − техническому;  − метрологическому;  − организационному (структура и функции эксплуатирующих подразделений, защита от ошибочных действий персонала);  − методическому (состав нормативно-технической документации) |
| Состав и содержание работ по созданию системы | Перечень стадий и этапов работ. Сроки исполнения. Состав организаций-исполнителей работ. Вид и порядок экспертизы технической документации. Программа обеспечения надежности. Программа метрологического обеспечения |
| Порядок контроля и приемки системы | Виды, состав, объем и методы испытаний системы. Общие требования к приемке работ по стадиям. Статус приемной комиссии |
| Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие | Преобразование входной информации к машиночитаемому виду. Изменения в объекте автоматизации. Сроки и порядок комплектования и обучения персонала |
| Требования к документированию | Перечень подлежащих разработке документов. Перечень документов на машинных носителях |
| Источники разработки | Документы и информационные материалы, на основании которых разрабатывается ТЗ и система |

**Задания для выполнения**

1. Разработать техническое задание на программный продукт (см. варианты заданий) в соответствии с ГОСТ 19.201-78 и ГОСТ 34.602—89
2. Оформить работу в соответствии с ГОСТ 19.106—78. При оформлении использовать MS Office или OpenOffice.org.
3. Сдать и защитить работу

**Содержание и оформление отчета по лабораторной работе**

Отчет по лабораторной работе должен состоять из:

1. Постановки задачи.

2. Технического задания на программный продукт. Отчет должен содержать титульный лист, аннотацию, содержание и основную часть, оформленную в соответствии с ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы» или ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению». В техническое задание в раздел календарного плана включить сетевой график/диаграмму Ганта, выполненную с помощью одного из инструментальных средств

Защита отчета по лабораторной работе заключается в предъявлении преподавателю полученных результатов (на экране монитора или у доски), демонстрации полученных навыков и ответах на вопросы преподавателя.

**Пример разработки технического задания на программный продукт**

Разработать техническое задание на разработку «Модуля автоматизированной системы оперативно-диспетчерского управления теплоснабжением корпусов Московского института».

|  |
| --- |
| «Утверждаю»  Зам. директора по УПР  \_\_\_\_\_\_\_\_ Иванов И.И.  «\_\_» \_\_\_\_\_\_\_201\_ г.  **Техническое задание**  на разработку «Модуля автоматизированной системы оперативно-диспетчерского управления теплоснабжением корпусов Московского института»  г.Москва, 2013 |

1. **Введение**

Работа выполняется в рамках проекта «Автоматизированная система оперативно- диспетчерского управления электротеплоснабжением корпусов Московского института».

**2. Основание для разработки**

2.1. Основанием для данной работы служит договор № 1234 от 10 марта 2013 г.

2.2. Наименование работы: «Модуль автоматизированной системы оперативно-диспетчерского управления теплоснабжением корпусов Московского института».

2.3. Исполнители: ОАО «Лаборатория создания программного обеспечения».

2.4. Соисполнители: нет.

**3. Назначение разработки**

Создание модуля для контроля и оперативной корректировки состояния основных параметров теплообеспечения корпусов Московского института.

**4. Технические требования**

4.1. Требования к функциональным характеристикам.

4.1.1. Состав выполняемых функций. Разрабатываемое ПО должно обеспечивать:

− сбор и анализ информации о расходовании тепла, горячей и холодной воды по данным теплосчетчиков SA-94 на всех тепловых выходах;

− сбор и анализ информации с устройств управления системами воздушного отопления и кондиционирования типа РТ1 и РТ2 (разработки кафедры СММЭ и ТЦ);

− предварительный анализ информации на предмет нахождения параметров в допустимых пределах и сигнализирование при выходе параметров за пределы допуска;

− выдачу рекомендаций по дальнейшей работе;

− отображение текущего состояния по набору параметров - циклически постоянно (режим работы круглосуточный), при сохранении периодичности контроля прочих пара метров;

− визуализацию информации по расходу теплоносителя: - текущую, аналогично показаниям счетчиков;

− с накоплением за прошедшие сутки, неделю, месяц - в виде почасового графика для информации за сутки и неделю;

− суточный расход — для информации за месяц.

Для устройств управления приточной вентиляцией текущая информация должна содержать номер приточной системы и все параметры, выдаваемые на собственный индикатор.

По отдельному запросу осуществляются внутренние настройки.

В конце отчетного периода система должна архивировать данные.

4.1.2. Организация входных и выходных данных.

Исходные данные в систему поступают в виде значений с датчиков, установленных в помещениях института. Эти значения отображаются на компьютере диспетчера. После анализа поступившей информации оператор диспетчерского пункта устанавливает необходимые параметры для устройств, регулирующих отопление и вентиляцию в помещениях. Возможна также автоматическая установка некоторых параметров для устройств регулирования.

Основной режим использования системы — ежедневная работа.

4.2. Требования к надежности.

Для обеспечения надежности необходимо проверять корректность получаемых данных с датчиков.

4.3. Условия эксплуатации и требования к составу и параметрам технических средств.

Для работы системы должен быть выделен ответственный оператор. Требования к составу и параметрам технических средств уточняются на этапе эскизного проектирования системы.

4.4. Требования к информационной и программной совместимости. 28 Программа должна работать на платформах Windows 98/ NT/2000.

4.5. Требования к транспортировке и хранению. Программа поставляется на лазерном носителе информации.

Программная документация поставляется в электронном и печатном виде.

4.6. Специальные требования:

− программное обеспечение должно иметь дружественный интерфейс, рассчитанный на пользователя (в плане компьютерной грамотности) квалификации;

− ввиду объемности проекта задачи предполагается решать поэтапно, при этом модули ПО, созданные в разное время, должны предполагать возможность наращивания системы и быть совместимы друг с другом, поэтому документация на принятое эксплуатационное ПО должна содержать полную информацию, необходимую для работы программистов с ним;

− язык программирования - по выбору исполнителя, должен обеспечивать возможность интеграции программного обеспечения с некоторыми видами периферийного оборудования (например, счетчик SA-94 и т. п.).

**5. Требования к программной документации**

Основными документами, регламентирующими разработку будущих программ, должны быть документы Единой Системы Программной Документации (ЕСПД): руководство пользователя, руководство администратора, описание применения.

**6. Технико-экономические показатели**

Эффективность системы определяется удобством использования системы для контроля и управления основными параметрами теплообеспечения помещений Московского института, а также экономической выгодой, полученной от внедрения аппаратно-программного комплекса.

**7. Порядок контроля и приемки**

После передачи Исполнителем отдельного функционального модуля программы Заказчику последний имеет право тестировать модуль в течение 7 дней. После тестирования Заказчик должен принять работу по данному этапу или в письменном виде изложить причину отказа принятия. В случае обоснованного отказа Исполнитель обязуется доработать модуль.

**8. Календарный план работ**

|  |  |  |  |
| --- | --- | --- | --- |
| № этапа | Название этапа | Сроки этапа | Чем заканчивается этап |
| 1 | Изучение предметной области. Проектирование системы. Разработка предложений по реализации системы | 01.02.200\_- 28.02.200\_ | Предложения по работе системы. Акт сдачи-приемки |
| 2 | Разработка программного модуля по сбору и анализу информации со счетчиков и устройств управления. Внедрение системы для одного из корпусов МИЭТ | 01.03.200\_- 31.08.200\_ | Программный комплекс, решающий поставленные задачи для пилотного корпуса МИЭТ. Акт сдачи-приемки |
| 3 | Тестирование и отладка модуля. Внедрение системы во всех корпусах МИЭТ | 01.09.200\_- 30.12.200\_ | Готовая система контроля теплообеспечения МИЭТ, установленная в диспетчерском пункте. Программная документация. Акт сдачи - приемки работ |

Руководитель работ Григорьева Г. Д.

**Лабораторная работа №4. Разработка архитектуры программного средства.**

**Цель работы:** реализация начальных этапов процесса разработки программного средства в соответствии с ГОСТ Р ИСО/МЭК 12207

1. **Краткие теоретические сведения**

При возникновении потребностей в заказе, приобретении, разработке, эксплуатации и сопровождении программ перед всеми сторонами, вовлеченными в жизненный цикл программного средства (ПС), возникает целый ряд вопросов, связанных с определением и детальным структурированием жизненного цикла (ЖЦ) ПС, с организационными и техническими правами и обязанностями сторон, с управлением ЖЦ и контролем за его реализацией. Одним из действенных инструментов для решения данных вопросов является использование унифицированных подходов, закрепленных в современных международных и российских стандартах.

Понятия «жизненный цикл системы» или «жизненный цикл программного средства» часто появляются в статьях и звучат в разговорах разработчиков, по крайней мере руководителей проектов и подразделений. Всем понятно, что относятся они к тому, что и в какой последовательности должно делаться при создании и эксплуатации систем. Но прежде чем две организации или два специалиста договорятся о том, что конкретно входит или не входит в ЖЦ, проходит значительное время. А позже вполне может обнаружиться, что эти двое (две «стороны») все-таки по-разному понимают, какие работы будут входить в ЖЦ, а какие - нет, какие проверки будут планироваться, когда и т. д. Естественно, общие принципы организации работ описаны давно, но что делать сторонам в конкретном проекте — это каждый раз приходится решать заново.

В стандартах, регламентирующих жизненный цикл программных средств, обобщаются опыт и результаты исследований множества специалистов и рекомендуются наиболее эффективные современные методы и процессы создания и развития комплексов программ. В результате таких обобщений оттачиваются технологические процессы и приемы разработки, а также методическая база для их автоматизации.

ЖЦ ПС в стандартах представляет собой набор этапов, частных работ и операций в последовательности их выполнения и взаимосвязи, регламентирующих ведение работ от подготовки технического задания до завершения испытаний ряда версий и окончания эксплуатации ПС или информационной системы (ИС).

Стандарты включают правила описания исходной информации, способов и методов выполнения операций, устанавливают правила контроля технологических процессов, требования к оформлению их результатов, а также регламентируют содержание технологических и эксплуатационных документов на комплексы программ. Они определяют организационную структуру коллектива, обеспечивают распределение и планирование заданий, а также контроль за ходом создания ПС.

Кроме вопросов выбора типа общего устройства ЖЦ есть проблемы с решением частных вопросов о включении или невключении в ЖЦ отдельных работ, очень важных для качества ПС и системы: что документировать при создании системы и ПС, какие работы должны будут гарантировать качество продукта, с какой степенью организационной независимости должны выполняться проверочные процедуры разных типов, чем будет обеспечиваться соответствие разрабатываемого ПС требованиям ко всей системе и соответствие ПС потребностям в системе.

Для того чтобы привнести порядок и понимание, общие для любых сторон, участвующих в ЖЦ систем и ПС, давно разрабатывались стандарты различных уровней утверждения - национальные и международные.

В России основы построения и использования профилей стандартов ЖЦ ПС заложены принятием в качестве базового стандарта ГОСТ Р ИСО/МЭК 12207. Данный документ введен в действие с 1 июля 2000 г., тесно взаимоувязан с рядом стандартов, принятых ранее, и с некоторыми стандартами, разрабатываемыми в данное время на основе прямого применения стандартов ИСО.

Актуальность стандарта ГОСТ Р ИСО/МЭК 12207 для современных условий настолько высока, что принятие в ISO его исходного, международного варианта вскоре вызвало самую положительную оценку российских экспертов. Был дан ряд рекомендаций, но его использованию в реальных условиях.

В данном стандарте **программное обеспечение** (ПО) или программный продукт определяется как набор компьютерных программ,процедур и,возможно, связанной с ними документации и данных.

**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.

В соответствии с ГОСТ Р ИСО/МЭК 12207 все процессы ЖЦ ПО разделены на три **группы:**

**1)Основные процессы:**

−приобретение;

−поставка;

−разработка;

−эксплуатация;

−сопровождение.

**2)Вспомогательные процессы:**

−документирование;

−управление конфигурацией;

−обеспечение качества;

−верификация;

−аттестация;

−совместная оценка;

−аудит;

−разрешение проблем.

**3)Организационные процессы:**

−управление;

−усовершенствование;

−создание инфраструктуры;

−обучение.

**Процесс разработки** предусматривает действия и задачи, выполняемые разработчиком, и включает следующие действия:

А) **Подготовительная работа** начинается с выбора модели ЖЦ ПО, соответствующей масштабу, значимости и сложности проекта. Действия и задачи процесса 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.

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

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

−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.

В) **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.

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

−трансформацию требований к ПО в архитектуру, определяющую на высоком уровне структуру ПО и состав ее компонентов;

−разработку и документирование программных интерфейсов ПО и баз данных;

−разработку предварительной версии пользовательской документации;

−разработку и документирование предварительных требований к тестам и планам интеграции ПО.

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

Г) **Детальное проектирование ПО** включает следующие задачи:

−описание компонентов и интерфейсов между ними на более низком уровне, достаточном для их последующего самостоятельного кодирования и тестирования;

−разработку и документирование детального проекта базы данных;

−обновление (при необходимости) пользовательской документации;

−разработку и документирование требований к тестам и плана тестирования компонентов ПО;

−обновление плана интеграции ПО.

Д) **Кодирование и тестирование ПО** охватывает задачи:

−разработку и документирование каждого компонента ПО и базы данных, а также совокупности тестовых процедур и данных для их тестирования;

−тестирование каждого компонента ПО и базы данных на соответствие предъявляемых к ним требованиям. Результаты тестирования компонентов должны быть документированы;

−обновление (при необходимости) пользовательской документации;

−обновление плана интеграции ПО.

Е) **Интеграция ПО** предусматривает сборку разработанных компонентов ПО в соответствии с планом интеграции и тестирование агрегированных компонентов. Для каждого из агрегированных компонентов разрабатываются наборы тестов и тестовые процедуры, предназначенные для проверки каждого из квалификационных требований при последующем квалификационном тестировании.

**Интеграция системы** заключается в сборке всех ее компонентов, включая ПО и оборудование. После интеграции система, в свою очередь, подвергается квалификационному тестированию на соответствие совокупности требований к ней. При этом также производится оформление и проверка полного комплекта документации на систему.

Ж) **Квалификационное тестирование** - это набор критериев и условий,

которые необходимо выполнить, чтобы квалифицировать программный продукт как соответствующий своим спецификациям и готовый к использованию в условиях эксплуатации.

**Квалификационное тестирование ПО** проводится разработчиком в присутствии заказчика (по возможности) для демонстрации того, что ПО удовлетворяет своим спецификациям и готово к использованию в условиях эксплуатации. Квалификационное тестирование выполняется для каждого компонента ПО по всем разделам требований при широком варьировании тестов. При этом также проверяются полнота технической и пользовательской документации и ее адекватность самим компонентам ПО.

З) **Установка ПО** осуществляется разработчиком в соответствии с планом в той среде и на том оборудовании, которые предусмотрены договором. В процессе установки проверяется работоспособность ПО и баз данных. Если устанавливаемое программное обеспечение заменяет существующую систему, разработчик должен обеспечить их параллельное функционирование в соответствии с договором.

И) **Приемка ПО** предусматривает оценку результатов квалификационного тестирования ПО и системы и документирование результатов оценки, которые проводятся заказчиком с помощью разработчика. Разработчик выполняет окончательную передачу ПО заказчику в соответствии с договором, обеспечивая при этом необходимое обучение и поддержку.

**2. Задание:** разработать проект архитектуры программного средства в соответствии с ГОСТ Р ИСО/МЭК 12207

**Алгоритм выполнения работы**

С целью реализации начальных этапов разработки ПС в соответствии с техническим заданием:

−выполнить подготовительную работу;

−провести анализ требований к ПС;

−выполнить проектирование архитектуры ПС на высоком уровне.

# **Лабораторная работа №5. Изучение работы в системе контроля версий**

**Цель работы:** ознакомиться с системой контроля версий Git.

1. **Краткие теоретические сведения**

**Git** — это система управления версиями, которая пришлась по душе практически всем — от разработчиков до дизайнеров. **GitHub** можно считать соцсетью для хранения кода. Это настоящая Мекка для технарей. Здесь вы можете попрактиковаться в разработке и придумать что-то свое, найти множество open-source проектов, передовых технологий, различных функций и дизайнов.

Для начала необходимо запомнить следующие терминальные команды:

git clone

git status

git add

git commit -m “ “

git push

Затем к ним добавим еще вот эти:

git init

git branch

git merge

git checkout

Эти команды вам пригодятся в случае, если вы будете работать с другими людьми или захотите внести какие-то изменения в проект и протестировать их до создания коммита.

Не лишней будет и вот такая команда:

git help

**Задание.**

**Шаг 1: Регистрация и установка**

Зайдите на [GitHub](https://github.com/?source=post_page---------------------------) и создайте свой аккаунт. В принципе, этим можно и ограничиться. При желании можете [установить Git](https://git-scm.com/downloads?source=post_page---------------------------). Но для работы с GitHub это вовсе не обязательно. Однако если вы планируете заниматься проектами на локальном компьютере, то установка вам все-таки нужна. Можете скачать установщик или [установить файлы через менеджер пакетов](https://gist.github.com/derhuerst/1b15ff4652a867391f03?source=post_page---------------------------).

Теперь перейдите в терминал, и начнем работу. Если хотите задать одно имя пользователя для **всех репозиториев** на компьютере, то напишите:

git config — global user.name “<ваше\_имя>”

замените <ваше\_имя> на свое имя в кавычках. Можете написать все, что угодно. Если хотите задать имя только для одного репозитория, то удалите из команды слово global.

Теперь напишите свой адрес электронной почты. Проследите, чтобы он совпадал с адресом, указанным при регистрации на GitHub.

git config — global user.email “<[адрес\_почты@email.com](mailto:%D0%B0%D0%B4%D1%80%D0%B5%D1%81_%D0%BF%D0%BE%D1%87%D1%82%D1%8B@email.com)>”

При желании можете скрыть свой электронный адрес. Это сделать несложно, подробнее написано [здесь](https://help.github.com/en/articles/blocking-command-line-pushes-that-expose-your-personal-email-address?source=post_page---------------------------). По сути, вам нужно проставить 2 галочки в своем GitHub-аккаунте.

**Теперь вы готовы к работе с Git на локальном компьютере.**

Начнем с создания нового репозитория на сайте GitHub. Вы также можете выполнить git init и создать новый репозиторий из директории проекта.

Репозиторий состоит из трех «деревьев». Первое «дерево» — это **рабочая директория,** в которой хранятся актуальные файлы. Второе — это **index** или область подготовленных файлов. А еще есть **head** — указатель на ваш последний коммит.

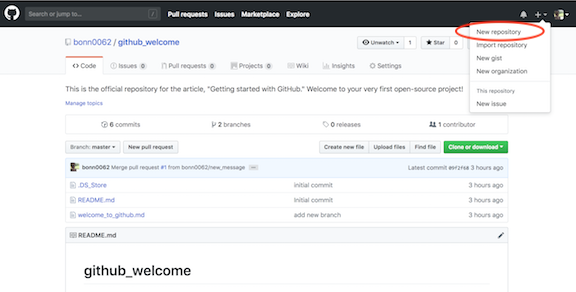
**Шаг 2. Создание репозитория**

Допустим, вы хотите создать новый репозиторий. Это место, где будет «жить» ваш проект. Если вы не хотите создавать новый репозиторий, то можете склонировать уже существующий. Именно так вы копируете чужой проект или берете нужную вам информацию для работы/учебы. Мы еще к этому вернемся, но чуть позже.

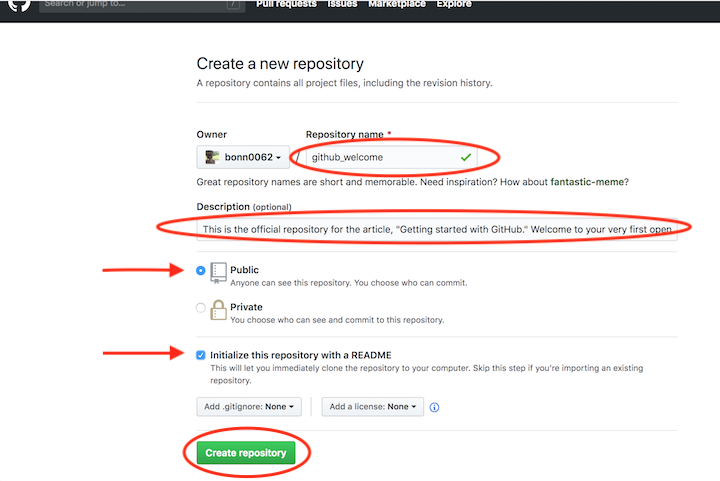
**Репозиторий** — это место, в котором вы систематизируете свой проект. Здесь вы храните файлы, папки, видео, изображения, блокноты Jupyter Notebook, наборы данных и т.д. Перед началом работы с Git необходимо инициализировать репозиторий для проекта и правильно его подготовить. Это можно сделать на сайте GitHub.

Лучше сразу добавлять в репозиторий **README**-файл с информацией о проекте. Это можно сделать в момент создания репозитория, поставив галочку в соответствующем поле.

* Перейдите на сайт GitHub. Нажмите на значок **+** в верхнем правом углу, а затем выберите **New repository**.
* Придумайте имя репозитория и добавьте короткое описание.
* Решите, будет ли этот репозиторий размещаться в открытом доступе или останется закрытым для просмотра.
* Нажмите **Initialize this repository with a README** для добавления README-файла. Настоятельно рекомендую снабжать все ваши проекты файлом-описанием, ведь README — это первая вещь, на которую люди обращают внимание при просмотре репозитория. К тому же, здесь можно разместить нужную информацию для понимания или запуска проекта.



Новый репозиторий



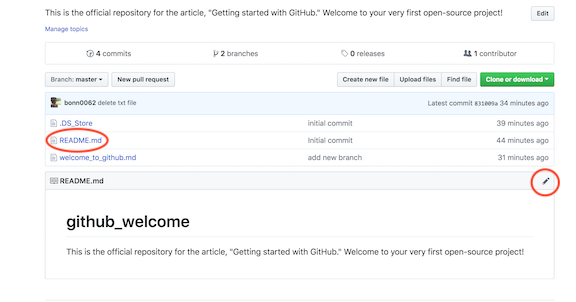
Создание нового репозитория

При желании можете уже сейчас начинать работать над проектом. Добавляйте файлы, вносите в них изменения и т.д. напрямую с сайта GitHub. Однако конечный результат подобной деятельности может вас немного огорчить.

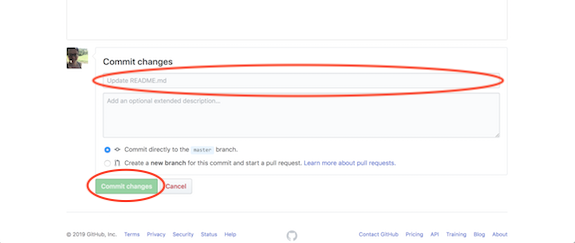
Вносить изменения в проект можно двумя способами. Вы можете изменять файлы/блокноты на компьютере либо делать это на сайте GitHub.

Допустим, вам захотелось подкорректировать README-файл на сайте GitHub.

* Для начала перейдите в ваш репозиторий.
* Для выбора файла кликните по его названию (например, кликните по **README.md** для перехода к файлу-описанию).
* В верхнем правом углу вы увидите иконку с карандашом. Нажмите на нее для внесения изменений.
* Напишите короткое сообщение, передающее суть изменений (и подробное описание, если сочтете это нужным).
* Нажмите кнопку **Commit changes**.



Изменение файла на GitHub

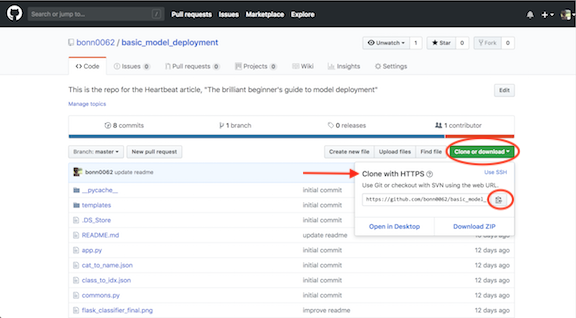


Подготовка коммита с изменениями

Вы успешно внесли изменения в README-файл своего нового репозитория! Обратите внимание на небольшую кнопку на картинке выше. Она позволяет создавать новую ветку этого коммита и добавлять Pull request. Запомните ее, скоро к ней вернемся.

Возможно, вы захотите клонировать свой новый репозиторий для дальнейшей работы с ним на локальном компьютере. Либо у вас уже есть существующий репозиторий, который вы хотели бы клонировать.

Для **клонирования репозитория** на компьютер перейдите в репозиторий на GitHub и нажмите большую зеленую кнопку под названием **Clone or download.** Проследите, чтобы появилась надпись **Clone with HTTPS**. Теперь нажмите на иконку буфера обмена для копирования-вставки (либо выделите ссылку и скопируйте ее).



Клонирование или скачивание репозитория

Откройте **терминал** и перейдите в директорию для копирования репозитория. Например, для перехода на **Рабочий стол** напечатайте вот это:

cd Desktop

Затем клонируйте туда репозиторий по следующей команде:

git clone <то,\_что\_вы\_только\_что\_скопировали>

Все просто! Не забудьте изменить информацию в угловых скобках на нужную вам. И удалите сами скобки **< >.**

*Если вы не очень хорошо ориентируетесь в терминале, то переход по директориям можно осуществлять через команду* ***cd****. Например, откройте терминал и напечатайте* ***ls*** *для отображения перечня доступных директорий. Вполне возможно, что в этом списке вы сразу увидите директорию* ***Desktop****. Либо напечатайте* ***cd******Desktop****. Далее выполните команду* ***git******clone*** *и склонируйте репозиторий на Рабочий стол.*

*Бывает и так, что вместо перечня расположений, вы видите различные имена пользователей. Тогда до того, как перейти в* ***Desktop****, вам потребуется выбрать нужного пользователя через команду* ***cd <пользователь>*** *(замените* ***<пользователь>*** *на нужное вам имя). Затем снова напечатайте* ***ls****, чтобы увидеть весь список. И вот теперь, увидев в списке* ***Desktop****, смело печатайте* ***cd******Desktop****. Сейчас уже можно выполнять* ***git******clone****!*

*Если вдруг в терминале вы захотите «откатиться» на шаг назад, то напишите* ***cd*** *..*

Новый GitHub-репозиторий, склонированный на рабочий стол, готов! Данная команда создает точную копию репозитория в вашей системе. Здесь вы сможете с ним работать, редактировать, индексировать изменения, создавать коммиты с изменениями и отправлять их на GitHub.

*Совсем не обязательно создавать репозиторий на Рабочем столе. Клонировать можно в любое место на компьютере. Команду* ***git******clone*** *можно выполнять и сразу после открытия терминала. Однако, если вы не очень любите копаться в папках на компьютере, то неплохо будет разместить проект на виду, то есть на Рабочем столе…*

Если хотите просто покопаться в каком-то проекте, то вместо клонирования можете сделать **форк** проекта на GitHub. Для этого нажмите кнопку **Fork** в верхнем правом углу сайта. Так вы добавите копию этого проекта в свои репозитории и сможете вносить туда любые изменения без вреда для оригинала.

Добавляем файлы в проект

Вот, чем мы займемся:

git status

git add

git commit -m “ “

git push

Но ничего сложного здесь нет!

Должно быть, у вас уже есть файлы, которые вы бы хотели разместить в новом репозитории. Отыщите их на компьютере и перетащите в новую папку репозитория на Рабочем столе.

Проверьте **статус** проекта.

Откройте терминал и перейдите в папку репозитория. Для проверки обновлений выполните:

git status

Если вы перетаскивали файлы в папку проекта, то потребуется обновить состояние репозитория. Добавлять файлы в репозиторий можно по одному:

git add <имя\_файла>

Либо все сразу:

git add — all

или даже:

git add .

Это ваши предлагаемые изменения. Операцию можно повторить с новыми файлами либо с уже существующими, но измененными. По сути, ничего нового в сам проект вы не добавляете. Вы всего лишь загружаете новые файлы и указываете Git на эти изменения.

Процесс создания коммитов с изменениями начинается с выполнения команды:

git commit -m “<сообщение\_о\_коммите>”

Коммиты изменений добавляются в **head** (указатель), а не в удаленный репозиторий. Не забудьте заменить текст в скобках и убрать **<>.** После внесения изменений создается снимок состояния репозитория, для чего используется команда **commit**. А через **–m** добавляется сообщение об этом снимке.

Сохраненные изменения и называются коммитом. При создании коммита вы добавляете сообщение о том, что именно менялось и почему. Так другие люди смогут лучше понять суть изменений.

Теперь ваши изменения сохранены в указателе локальной копии проекта. Для отправки изменений на удаленный репозиторий выполните команду:

git push

Тем самым вы отправляете изменения напрямую в репозиторий. Если вы работаете на локальном компьютере и хотите, чтобы коммиты отображались в онлайн, то необходимо своевременно отправлять эти изменения на GitHub по команде **git push**.

Актуальность версии можно проверить в любое время через команду **git status.**

Итог: у вас есть свой GitHub репозиторий, вы научились добавлять и изменять в нем файлы.

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.

**Лабораторная работа №6. Построение диаграмм Вариантов использования и Последовательности**

**Цель работы**: научиться создавать диаграмму вариантов использования и последовательности.

**Теоретические сведения.**

**Диаграмма Вариантов использования.**

*Диаграмма прецедентов* (диаграмма вариантов использования) в [UML](https://ru.wikipedia.org/wiki/UML) — диаграмма, отражающая отношения между [актёрами](https://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%82%D1%91%D1%80_%28UML%29) и [прецедентами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82_%28UML%29) и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

*Прецедент* — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних [требований](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E) к системе.

Основное назначение диаграммы — описание функциональности и поведения, позволяющее [заказчику](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%B0%D0%B7%D1%87%D0%B8%D0%BA), [конечному пользователю](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C) и [разработчику](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82) совместно обсуждать проектируемую или существующую [систему](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%28%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%29).

При моделировании системы с помощью диаграммы прецедентов [системный аналитик](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D1%82%D0%B8%D0%BA) стремится:

* чётко отделить систему от её окружения;
* определить действующих лиц (актёров), их взаимодействие с системой и ожидаемый функционал системы;
* определить в глоссарии предметной области понятия, относящиеся к детальному описанию функционала системы (то есть, прецедентов).

Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются (для них составляется другой документ).

Для отражения модели прецедентов на диаграмме используются:

* рамки системы ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) system boundary) — прямоугольник с названием в верхней части и эллипсами (прецедентами) внутри. Часто может быть опущен без потери полезной информации,
* *актёр* (англ. actor) — стилизованный человечек, обозначающий набор ролей пользователя (понимается в широком смысле: человек, внешняя сущность, класс, другая система), взаимодействующего с некоторой сущностью (системой, подсистемой, классом). Актёры не могут быть связаны друг с другом (за исключением отношений обобщения/наследования),
* *прецедент* — эллипс с надписью, обозначающий выполняемые системой действия (могут включать возможные варианты), приводящие к наблюдаемым актёрами результатам. Надпись может быть именем или описанием (с точки зрения актёров) того, «что» делает система (а не «как»). Имя прецедента связано с непрерываемым (атомарным) сценарием — конкретной последовательностью действий, иллюстрирующей поведение. В ходе сценария актёры обмениваются с системой сообщениями. Сценарий может быть приведён на диаграмме прецедентов в виде UML-комментария. С одним прецедентом может быть связано несколько различных сценариев.

Часть дублирующийся информации в модели прецедентов можно устранить указанием связей между прецедентами:

* *обобщение прецедента* — стрелка с незакрашенным треугольником (треугольник ставится у более общего прецедента),
* *включение прецедента* — пунктирная стрелка со [стереотипом](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D1%80%D0%B5%D0%BE%D1%82%D0%B8%D0%BF_%28UML%29) «include»,
* *расширение прецедента* — пунктирная стрелка со стереотипом «extend» (стрелка входит в расширяемый прецедент, в дополнительном разделе которого может быть указана точка расширения и, возможно в виде комментария, условие расширения)

При работе с вариантами использования важно помнить несколько простых правил:

* каждый прецедент относится как минимум к одному действующему лицу;
* каждый прецедент имеет инициатора;
* каждый прецедент приводит к соответствующему результату.

Диаграмма вариантов использования является высокоуровневым концептуальным представлением модели, поэтому она не должна содержать слишком много вариантов использования и актеров. В последующем построенная диаграмма может быть изменена посредством добавления новых элементов, таких как варианты использования и актеры, или их удаления.

**Технология работы**

Для разработки диаграммы вариантов использования модели в среде IBM Rational Rose необходимо активизировать соответствующую диаграмму в окне диаграммы. Это можно сделать следующими способами:

* раскрыть представление вариантов использования **Use Case View** в браузере проекта и дважды щелкнуть на пиктограмме **Main** (Главная);
* с помощью операции главного меню **Browse>Use Case Diagram** (Браузер>Диаграмма вариантов использования).

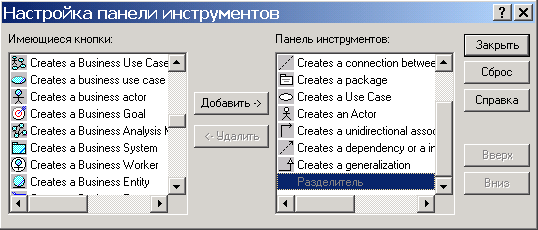
При этом появляется новое окно с чистым рабочим листом диаграммы вариантов использования и специальная панель инструментов, содержащая кнопки с изображением графических элементов, необходимых для разработки диаграммы вариантов использования. Назначение отдельных кнопок данной панели можно узнать также из всплывающих подсказок, которые появляются, если подвести и задержать на некоторое время указатель мыши над той или иной кнопкой (табл.1).

Таблица 1. Назначение кнопок специальной панели инструментов для диаграммы вариантов использования

|  |  |  |
| --- | --- | --- |
| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_1.png | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_2.png | Text Box | Добавляет на диаграмму текстовую область |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_3.png | Note | Добавляет на диаграмму примечание |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_4.png | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_5.png | Package | Добавляет на диаграмму пакет |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_6.png | Use Case | Добавляет на диаграмму вариант использования |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_7.png | Actor | Добавляет на диаграмму актера |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_8.png | Unidirectional Association | Добавляет на диаграмму направленную ассоциацию |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_9.png | Dependency or Instantiates | Добавляет на диаграмму отношение зависимости |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/3/files/3_10.png | Generalization | Добавляет на диаграмму отношение обобщения |

На специальной панели инструментов по умолчанию присутствует только часть кнопок с пиктограммами элементов, которые могут быть использованы для построения диаграммы. Добавить кнопки с пиктограммами других графических элементов, например, таких как бизнес-вариант использования (business use case), бизнес-актер (business actor), сотрудник (business worker), или удалить ненужные кнопки можно с помощью настройки специальной панели инструментов.

Открыть диалоговое окно настройки специальных панелей инструментов для диаграмм в среде IBM Rational Rose можно с помощью операции главного меню: **Tools>Options** (Инструменты>Параметры), раскрыв вкладку **Toolbars** (Панели инструментов) и нажав соответствующую кнопку (например, **Use Case diagram** ) в группе опций **Customize Toolbars** (Настройка панелей инструментов). Это окно настройки также можно открыть с помощью операции контекстного меню **Customize** (Настройка) при позиционировании курсора на специальной панели инструментов (рис.1).

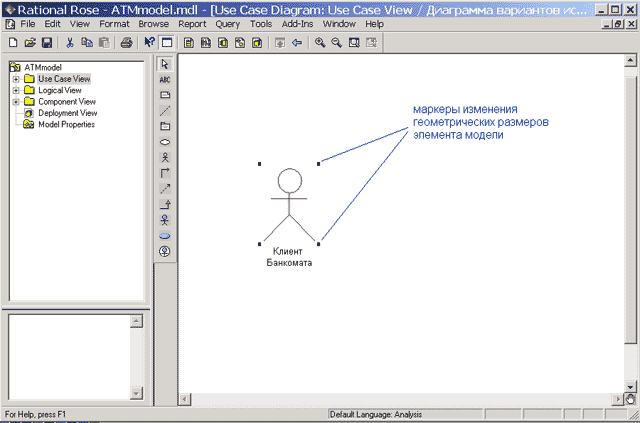


**Рис. 1.** Диалоговое окно настройки специальной панели инструментов для диаграммы вариантов использования

Для добавления необходимых кнопок на панель следует выделить их в левом окне со списком пиктограмм графических элементов, после чего нажать кнопку **Добавить** в центре диалогового окна. Для удаления ненужных кнопок с панели инструментов следует выделить их в правом окне со списком пиктограмм графических элементов, после чего нажать кнопку **Удалить** в центре диалогового окна. Для восстановления набора пиктограмм по умолчанию можно нажать кнопку **Сброс**. После настройки специальной панели инструментов соответствующее окно следует закрыть нажатием на кнопку **Закрыть**.

**1.Добавление актера на диаграмму вариантов использования и редактирование его свойств**

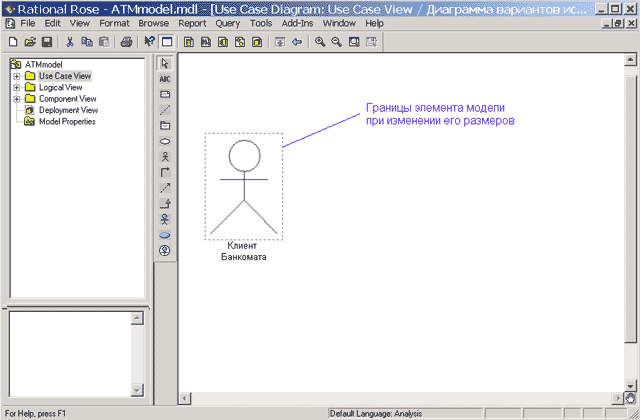
Для добавления актера на диаграмму варианта использования нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы актера на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение актера с маркерами изменения его геометрических размеров и предложенным программой именем по умолчанию NewClass. Для разрабатываемой модели банкомата предложенное программой имя актера следует изменить на Клиент Банкомата (рис. 2).



**Рис. 2.** Диаграмма вариантов использования после добавления на нее актера

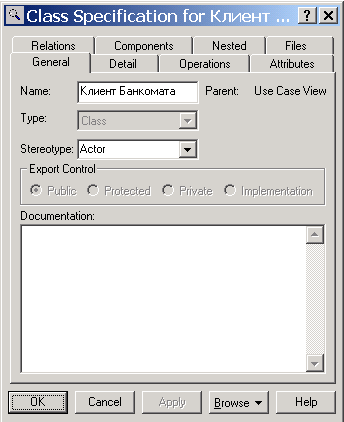
Чтобы изменить расположение изображения графического элемента модели, следует щелчком левой кнопки мыши выделить его в рабочей области диаграммы, и, не отпуская левой кнопки, переместить в нужное место диаграммы. При этом выделенный элемент визуально отличается от остальных наличием маркеров изменения его геометрических размеров в форме небольших черных квадратов. Более точное перемещение элемента можно осуществить с помощью стрелок: Описание: \leftarrow, Описание: \uparrow, Описание: \rightarrow, Описание: \downarrowна клавиатуре.

Чтобы изменить графические размеры изображения элемента модели, прежде всего, следует щелчком левой кнопки мыши выделить его в рабочей области диаграммы. Далее необходимо подвести указатель мыши к нужному маркеру геометрических размеров элемента и нажать левую кнопку мыши. В результате этих действий появится пунктирный прямоугольник, изображающий границы выбранного геометрического элемента. После чего, не отпуская левой кнопки мыши, следует диагонально изменить размеры этого прямоугольника нужным образом (рис. 3).



**Рис. 3.** Диаграмма вариантов использования при изменении графических размеров актера

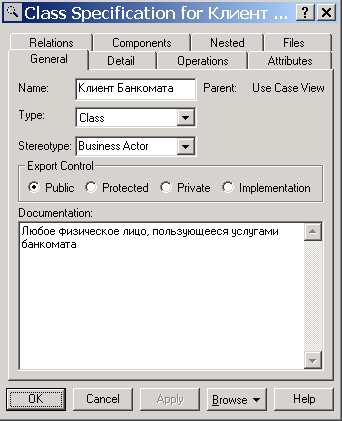
Имя размещенного на диаграмму элемента разработчик может изменить либо сразу после добавления элемента на диаграмму, либо в ходе последующей работы над проектом. Для любого графического элемента модели по щелчку правой кнопкой мыши на выбранном элементе вызывается контекстное меню данного элемента, среди операций которого имеется пункт **Open Specification** (Открыть спецификацию). В этом случае появляется дополнительное диалоговое окно со специальными вкладками, в поля ввода которых можно занести всю информацию по данному элементу. Для добавленного актера Клиент Банкомата окно спецификации свойств выглядит следующим образом (рис. 4).



**Рис. 4.** Диалоговое окно спецификации свойств актера Клиент Банкомата

Следует отметить, что открыть диалоговое окно спецификации свойств любого элемента модели можно также двойным щелчком левой кнопкой мыши на графическом изображении этого элемента на диаграмме. Хотя в среде IBM Rational Rose актер является классом, для него некорректно специфицировать атрибуты и операции, поскольку актер является внешней по отношению к разрабатываемой системе сущностью.

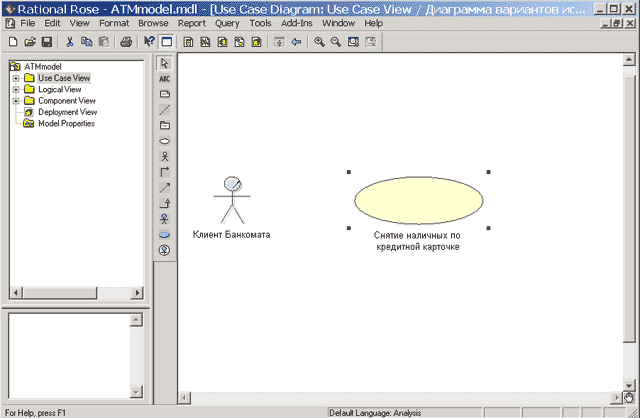
Для актера Клиент Банкомата можно уточнить его назначение в модели. С этой целью следует изменить его стереотип и добавить текст документации. Для изменения стереотипа во вложенном списке **Stereotype** нужно выбрать строку **Business Actor** ( бизнес-актер ). Для добавления текста документации в секцию **Documentation** следует ввести текст: "Любое физическое лицо, пользующееся услугами банкомата" и нажать кнопку **Apply** (Применить) или **OK**. После изменения данных свойств актера Клиент Банкомата окно спецификации свойств будет выглядеть следующим образом (рис.5).



**Рис. 5.** Диалоговое окно спецификации свойств после изменения стереотипа и добавления текста документации для актера Клиент Банкомата

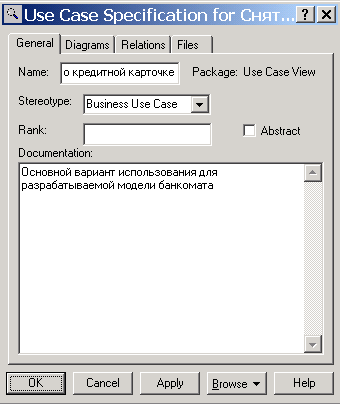
**2.Добавление и редактирование варианта использования**

Для добавления варианта использования на диаграмму нужно с помощью левой кнопки мыши нажать кнопку с изображением варианта использования на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте диаграммы. На диаграмме появится изображение варианта использования с маркерами изменения его геометрических размеров и предложенным программой именем по умолчанию NewUseCase. Для разрабатываемой модели банкомата предложенное программой имя варианта использования следует изменить на Снятие наличных по кредитной карточке (рис. 6).



**Рис. 6.** Диаграмма вариантов использования после добавления на нее варианта использования

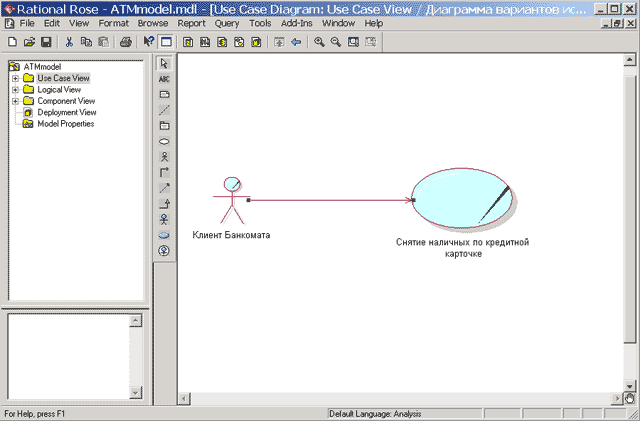
Для уточнения свойств данного варианта использования следует открыть диалоговое окно спецификации его свойств, например, с помощью двойного щелчка левой кнопкой мыши на изображении этого элемента на диаграмме. Для изменения стереотипа во вложенном списке **Stereotype** нужно выбрать строку **Business Use Case**. Для добавления текста документации в секцию **Documentation** следует ввести текст: "Основной вариант использования для разрабатываемой модели банкомата" и нажать кнопку **Apply** (Применить) или **OK**. После изменения данных свойств варианта использования окно спецификации его свойств будет выглядеть следующим образом (рис. 3.7).



**Рис. 7.** Диалоговое окно спецификации свойств варианта использования Снятие наличных по кредитной карточке

**3.Добавление ассоциации**

Для добавления ассоциации между актером и вариантом использования на диаграмму нужно с помощью левой кнопки мыши нажать на специальной панели инструментов кнопку с изображением пиктограммы направленной ассоциации, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении актера на диаграмме и отпустить ее на изображении варианта использования. В результате этих действий на диаграмме появится изображение ассоциации, соединяющей актера с вариантом использования (рис. 8).

  
**Рис. 8.** Диаграмма вариантов использования после добавления на нее направленной ассоциации

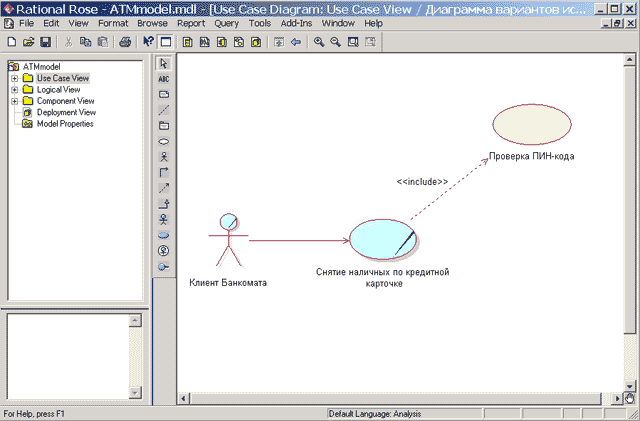
При необходимости можно сделать направленную ассоциацию ненаправленной, для чего следует воспользоваться диалоговым окном свойств ассоциации. Открыть это окно можно, например, двойным щелчком на изображении линии ассоциации на диаграмме, после чего убрать отметку строки выбора **Navigable** (Навигация) на вкладке **Role A Detail** (Детальные свойства концевой точки ассоциации А).

**4.Добавление отношения зависимости и редактирование его свойств**

Для добавления отношения зависимости между двумя вариантами использования на диаграмму необходимо предварительно рассмотренным выше способом добавить второй вариант использования с именем Проверка ПИН-кода. После этого с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы зависимости на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении варианта использования Снятие наличных по кредитной карточке и отпустить ее на изображении варианта использования Проверка ПИН-кода. В результате этих действий на диаграмме появится изображение отношения зависимости, которое соединяет два выбранных варианта использования.

Поскольку вариант использования Проверка ПИН-кода выполняется всегда, для добавленного отношения зависимости дополнительно следует указать текстовый стереотип <<include>>. Выполнить это можно уже известным способом с помощью диалогового окна спецификации свойств этого отношения и выбора нужного стереотипа из предлагаемого списка.

После задания для данного отношения зависимости стереотипа <<include>> текст этого стереотипа в угловых скобках появится рядом с изображением пунктирной линии зависимости, связывающей соответствующие варианты использования (рис. 3.9). С целью лучшей визуализации диаграммы текстовую область стереотипа можно переместить в нужное место диаграммы. Выполнить это можно с помощью общего способа перемещения графических элементов модели, который был рассмотрен ранее в этой лекции применительно к актеру Клиент Банкомата.



**Рис. 9.** Диаграмма вариантов использования после добавления на нее отношения зависимости

Аналогичным образом могут быть добавлены на диаграмму вариантов использования отношения зависимости со стереотипом <<extend>>, которые применяются для моделирования исключений при выполнении отдельных вариантов использования.

К отдельному варианту использования можно добавить текстовый файл с описанием сценария его выполнения. Для этого необходимо выделить этот вариант использования в браузере проекта и выполнить операцию контекстное меню: **New>File** (Новый>Файл). В результате этого будет вызвано стандартное окно открытия файла, в котором необходимо задать имя предварительно созданного с помощью офисной программы MS Word добавляемого файла. После нажатия кнопки **Открыть** пиктограмма добавленного файла появится в браузере проекта ниже соответствующего варианта использования. В последующем можно вернуться к редактированию этого файла сценария, выполнив двойной щелчок на этой пиктограмме. При этом файл сценария будет открыт в соответствующем приложении - в текстовом процессоре MS Word.

Для удаления любого графического элемента с диаграммы его следует выделить на диаграмме и нажать клавишу **Delete** на клавиатуре. При этом выделенный элемент будет удален с активной диаграммы, но не из модели. Для удаления элемента не только из диаграммы, но и из модели проекта необходимо выделить удаляемый элемент на диаграмме и воспользоваться операцией главного меню **Edit>Delete from Model** (Редактирование>Удалить из модели). Для этой же цели служит комбинация клавиш быстрого доступа: **Ctrl+D**.

При работе с отношениями на диаграмме вариантов использования следует помнить о назначении соответствующих отношений в нотации языка UML. Речь идет о том, что если для двух элементов выбранный вид отношения не является допустимым, то в большинстве случаев программа IBM Rational Rose сообщит об этом разработчику, и соответствующая линия связи не будет добавлена на диаграмму.

После окончания сеанса работы над проектом выполненную работу необходимо сохранить в файле проекта с расширением ".MDL". Это можно сделать через меню **File>Save** (Файл>Сохранить) или **File>Save As** (Файл>Сохранить как). При этом вся информация о проекте, включая диаграммы и спецификации элементов, будет сохранена в одном файле.

**Диаграмма Последовательности**

Диаграмма последовательности является другой формой визуализации взаимодействия в модели и, как и диаграмма кооперации, оперирует объектами и сообщениями. Особенность работы в среде IBM Rational Rose заключается в том, что этот вид канонической диаграммы может быть создан автоматически после построения диаграммы кооперации и нажатия клавиши <F5>. С помощью этой же клавиши осуществляется переключение между диаграммами последовательности и кооперации в модели.

**Технология работы**

Однако в отдельных случаях бывает удобно начать построение диаграмм взаимодействия с диаграммы последовательности. В этом случае активизировать рабочее окно диаграммы последовательности можно несколькими способами:

• Щелкнуть на кнопке с изображением диаграммы взаимодействия на стандартной панели инструментов и выбрать для построения диаграмму последовательности.

• Выполнить операцию главного меню: Browse Interaction Diagram (Браузер Диаграмма взаимодействия) и выбрать для построения новую диаграмму последовательности.

• Выполнить операцию контекстного меню: New Sequence Diagram (Новая Диаграмма последовательности) для логического представления или представления вариантов использования в браузере проекта.

При этом появляется новое окно с чистым рабочим листом диаграммы классов и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы последовательности (табл. 1). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 1. Назначение кнопок специальной панели инструментов диаграммы последовательности

|  |  |  |
| --- | --- | --- |
| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_1.png | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_2.png | Text Box | Добавляет на диаграмму текстовую область |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_3.png | Note | Добавляет на диаграмму примечание |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_4.png | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_5.png | Object | Добавляет на диаграмму объект |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_6.png | Object Message | Добавляет на диаграмму простое сообщение |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_7.png | Message To Self | Добавляет на диаграмму рефлексивное сообщение |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_8.png | Return Message | Добавляет на диаграмму сообщение типа возврата из вызова процедуры |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_9.png | Destruction Marker | Добавляет на диаграмму символ уничтожения объекта |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_10.png | Procedure Call | Добавляет на диаграмму сообщение типа вызова процедуры (по умолчанию отсутствует) |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/8/files/8_11.jpg | Asynchronous Message | Добавляет на диаграмму асинхронное сообщение (по умолчанию отсутствует) |

На специальной панели инструментов по умолчанию присутствует практически все пиктограммы элементов, которые могут быть использованы для построения диаграммы последовательности. Из дополнительных пиктограмм графических элементов на специальную панель инструментов можно добавить лишь сообщение типа вызова процедуры и асинхронное сообщение (последняя строка табл. 1). Относительно изображения асинхронного сообщения в форме полустрелки следует заметить, что хотя в версии языка UML 1.5 этот элемент отсутствует, в среде IBM Rational Rose возможно изобразить этот тип сообщений в форме специального графического стереотипа.

**1.Добавление объекта на диаграмму последовательности и редактирование его свойств**

Добавить объект на диаграмму последовательности можно как стандартным образом с помощью соответствующей кнопки на специальной панели инструментов, так и более удобным способом - с помощью перетаскивания изображения пиктограммы класса из браузера на свободное место рабочего листа диаграммы последовательности.

В результате этих действий на диаграмме последовательности появится изображение объекта с именем класса, маркерами изменения его геометрических размеров и вертикальной пунктирной линией, означающей линию жизни этого объекта (рис. 10).

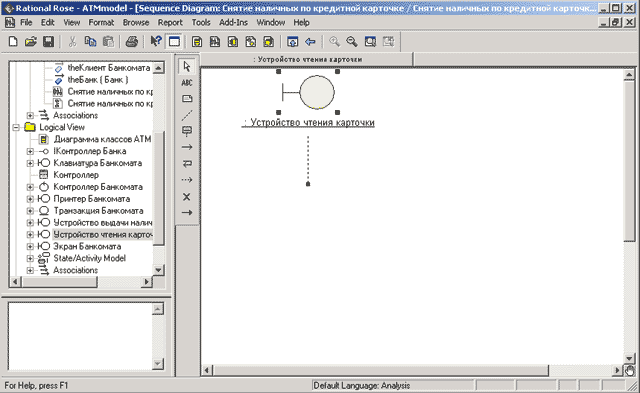


Рис. 10. Диаграмма последовательности после добавления анонимного объекта класса Устройство чтения карточки

Так же как и для диаграммы кооперации, для диаграммы последовательности каждый добавляемый объект по умолчанию считается анонимным. При необходимости можно задать собственное имя объекта, для чего уже известным способом (например, двойным щелчком на изображении объекта на диаграмме) следует вызвать диалоговое окно свойств объекта, которое аналогично объектам диаграммы кооперации.

**2.Добавление сообщения на диаграмму последовательности и редактирование его свойств**

Для добавления сообщения между предварительно размещенными на диаграмме объектами нужно с помощью левой кнопки мыши нажать кнопку с изображением сообщения на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении линии жизни одного объекта на диаграмме и отпустить ее на изображении линии жизни второго объекта.

В результате этих действий на диаграмме появится изображение сообщения, передаваемого, например, от экземпляра актера Клиент Банкомата объекту класса Устройство чтения карточки. Поскольку кнопка с изображением актера отсутствует на специальной панели инструментов диаграммы последовательности, соответствующий объект следует предварительно поместить на диаграмму способом перетаскивания пиктограммы актера из браузера проекта. При этом изображение линии жизни у соответствующей пары объектов изменится на изображение фокуса управления (рис.11).

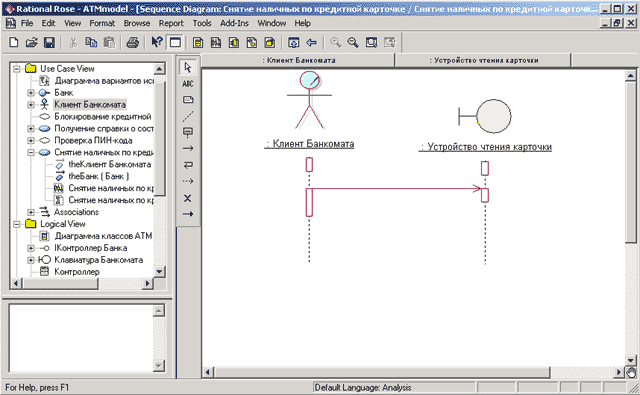


Рис. 11. Диаграмма последовательности после добавления сообщения от экземпляра актера Клиент Банкомата к объекту класса Устройства чтения карточки

Для спецификации свойств добавленного сообщения предназначено специальное окно, которое можно открыть двойным щелчком на изображении сообщения на диаграмме последовательности. Имя сообщения можно выбрать на вкладке General (Общие) из выпадающего списка операций соответствующего класса-приемника (рис. 12).

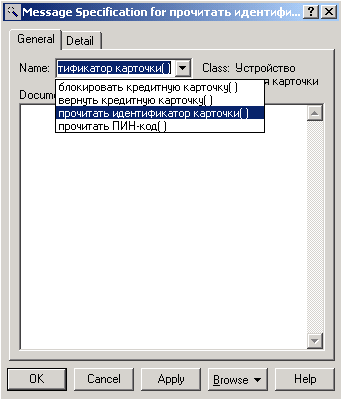


Рис. 12. Диалоговое окно спецификации свойств сообщения

Имя сообщения можно выбрать также из контекстного меню сообщения, в котором перечислены все операции класса-приемника данного сообщения (рис. 13). При необходимости в контекстном меню можно задать новую операцию, в этом случае следует выбрать строку <new operation>. При этом откроется диалоговое окно спецификации свойств новой операции класса-приемника.

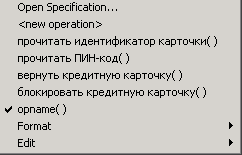


Рис.13. Контекстное меню сообщения на диаграмме последовательности

Для рассматриваемой модели банкомата в качестве имени первого сообщения следует выбрать операцию прочитать идентификатор карточки(). После выбора операции для данного сообщения следует нажать кнопку Apply или OK, в результате чего имя сообщения будет изображено на диаграмме последовательности рядом с линией сообщения (рис. 14).

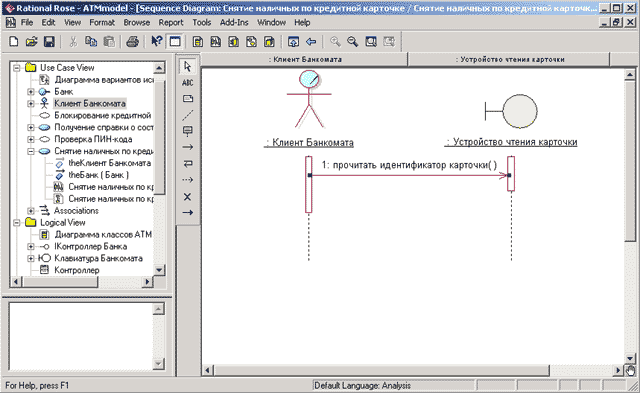


Рис. 14. Диаграмма последовательности после добавления сообщения от экземпляра актера Клиент Банкомата к объекту класса Устройство чтения карточки

Построение диаграммы последовательности сводится к добавлению и редактированию свойств отдельных объектов и сообщений. Доступ к окну спецификации свойств соответствующих элементов возможен также либо через контекстное меню, либо с помощью операции главного меню Browse Specification (Обзор Спецификация). При добавлении сообщений на диаграмму последовательности они получают по умолчанию свой номер в общей последовательности сообщений.

Следует заметить, что по умолчанию нумерация сообщений на диаграмме последовательности может быть отключена. При необходимости показать номера сообщений следует выполнить операцию главного меню: Tools Options (Инструменты Параметры), открыть вкладку Diagram (Диаграмма) и выставить отметку выбора строки Sequence numbering (Нумерация сообщений на диаграмме последовательности) в группе свойств Display (рис. 15).

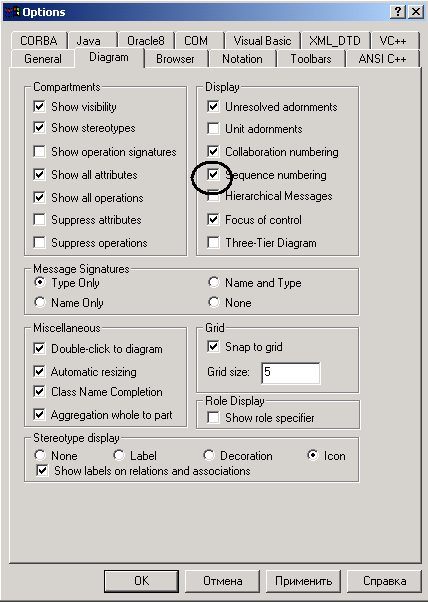


Рис. 15. Диалоговое окно спецификации свойств модели

Это же окно спецификации свойств модели можно открыть с помощью операции главного меню: View Toolbars Configure (Вид Панели инструментов Настроить),

Для детальной спецификации свойств сообщений на диаграмме последовательности можно использовать также группу свойств Synchronization (Синхронизация) и Frequency (Частота), доступные для выбора на вкладке Detail (Подробно) окна спецификации сообщения. При изменении способа синхронизации передаваемого сообщения изменяется графическое изображение стрелки соответствующего сообщения.

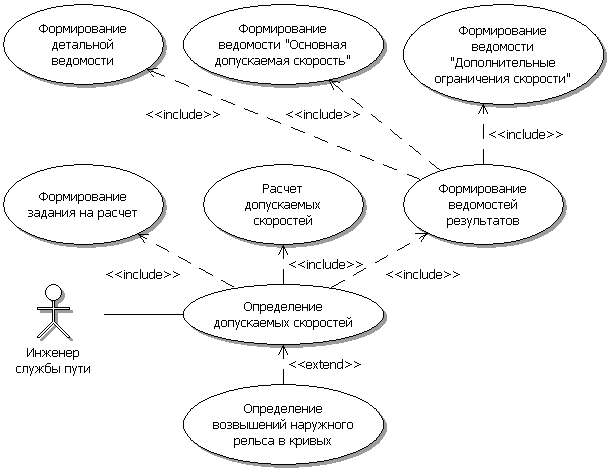


Рис.16. Пример Диаграммы Вариантов использования

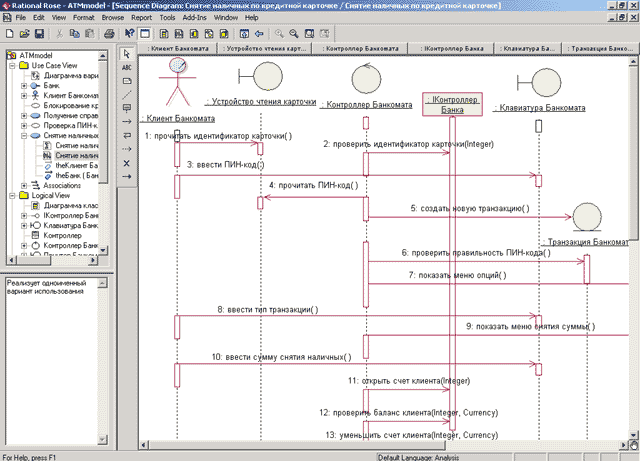


Рис.17. Пример Диаграммы Последовательности

**Задание:**

Изучить теоретические сведения по диаграммам Варианта использования и Последовательности. Создать данные диаграммы по своей предметной области, ссылая на примеры диаграмм (рис.16-17).

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.

**Лабораторная работа №7. Построение диаграмм Кооперации и Развертывания**

**Цель работы**: научиться создавать диаграмму кооперации и развертывания.

**Теоретические сведения**

*Диаграмма кооперации* является разновидностью диаграммы взаимодействия, и в контексте языка UML описывает динамический аспект взаимодействия объектов при реализации отдельных вариантов использования.

***Технология работы:***

Активизировать рабочее окно диаграммы кооперации в программе IBM Rational Rose можно несколькими способами:

* Щелкнуть на кнопке с изображением диаграммы взаимодействия на стандартной панели инструментов и выбрать для построения новую диаграмму кооперации.
* Выполнить операцию главного меню: **Browse Описание: \toInteraction Diagram** (Браузер Описание: \toДиаграмма взаимодействия) и выбрать для построения новую диаграмму кооперации.
* Выполнить операцию контекстного меню: **New Описание: \toCollaboration Diagram** (Новая Описание: \toДиаграмма кооперации) для логического представления или представления вариантов использования в браузере проекта.

При этом появляется новое окно с чистым рабочим листом диаграммы кооперации и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы кооперации (табл. 1). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 7.1. Назначение кнопок специальной панели инструментов диаграммы кооперации

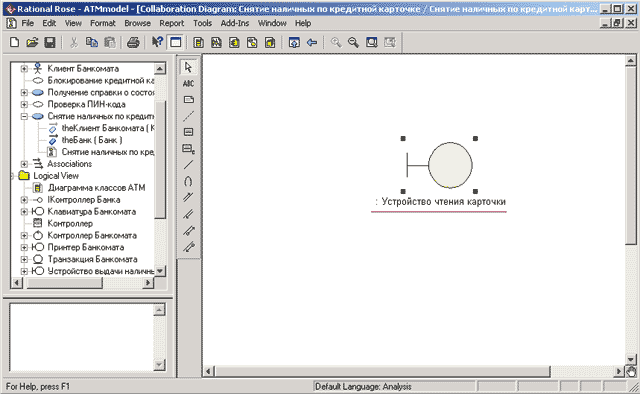
|  |  |  |
| --- | --- | --- |
| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_1.png | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_2.png | Text Box | Добавляет на диаграмму текстовую область |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_3.png | Note | Добавляет на диаграмму примечание |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_4.png | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_5.png | Object | Добавляет на диаграмму объект |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_6.png | Class Instance | Добавляет на диаграмму экземпляр класса |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_7.png | Object Link | Добавляет на диаграмму связь |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_8.png | Link To Self | Добавляет на диаграмму рефлексивную связь |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_9.png | Link Message | Добавляет на связь диаграммы прямое сообщение |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_10.png | Reverse Link Message | Добавляет на связь диаграммы обратное сообщение |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_11.png | Data Token | Добавляет на связь диаграммы элемент прямого потока данных |
| Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_12.png | Reverse Data Token | Добавляет на связь диаграммы элемент обратного потока данных |

На специальной панели инструментов по умолчанию присутствуют практически все кнопки с пиктограммами элементов, которые могут быть использованы для построения диаграммы. В данной лекции в качестве примера рассматривается процесс построения диаграммы кооперации, которая представляет собой реализацию варианта использования Снятие наличных по кредитной карточке применительно к разрабатываемому проекту системы управления банкоматом. В модели данная диаграмма кооперации соответствует этому варианту использования и может быть размещена в представлении вариантов использования ( **Use Case View** ). После активизации новой диаграммы кооперации одним из описанных выше способов следует в качестве имени данной диаграммы задать: Снятие наличных по кредитной карточке.

В общем случае работа с диаграммой кооперации состоит в добавлении объектов, связей и сообщений, а также редактировании их свойств. При этом изменения, вносимые в диаграмму кооперации, автоматически вносятся в диаграмму последовательности, что можно увидеть в любой момент, активизировав последнюю нажатием клавиши <F5>.

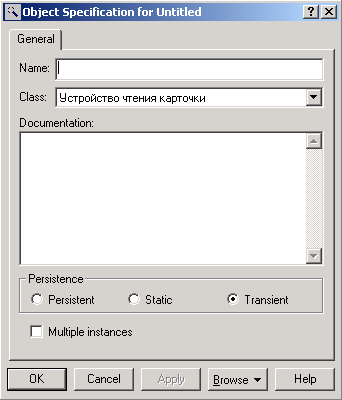
**1.Добавление объекта на диаграмму кооперации и редактирование его свойств**

Добавить объект на диаграмму кооперации можно стандартным образом с помощью соответствующей кнопки на специальной панели инструментов. Однако, в случае наличия построенной ранее диаграммы классов, более удобным представляется следующий способ. В браузере проекта выделить необходимый класс и, удерживая нажатой левую кнопку мыши, перетащить изображение пиктограммы класса из браузера на свободное место рабочего листа диаграммы кооперации. В результате этих действий на диаграмме кооперации появится изображение объекта с именем класса и маркерами изменения его геометрических размеров (рис. 7.1).



**Рис. 7.1.** Диаграмма кооперации после добавления на нее анонимного объекта класса Устройство чтения карточки

По умолчанию каждый добавляемый объект считается анонимным. При необходимости можно задать собственное имя объекта, для чего двойным щелчком на изображении объекта на диаграмме кооперации следует вызвать диалоговое окно свойств этого объекта (рис. 7.2).



**Рис. 7.2.** Диалоговое окно спецификации свойств объекта класса Устройство чтения карточки

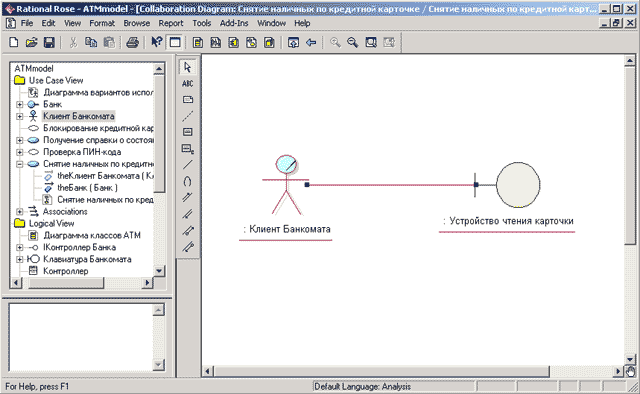
Как видно из рассмотрения этого окна свойств, для объекта выбранного класса можно задавать: собственное имя объекта, особенности его реализации и множественность экземпляров.

Группа свойств **Persistence** (Устойчивость) предназначена для спецификации устойчивости объектов соответствующего класса. При этом свойство **Persistent** (Устойчивый) означает, что информация об объектах данного класса должна быть сохранена в системе некоторым подходящим способом. Свойство **Static** (Статический) означает, что соответствующий объект сохраняется в памяти компьютера в течение всего времени работы программного приложения. Свойство **Transient** (Временный) соответствующий объект хранится в памяти компьютера в течение короткого времени, необходимого только для выполнения его операций. Применительно к рассматриваемой для объекта класса Устройство чтения карточки модели следует выбрать свойство **Persistent**.

При необходимости можно представить объект в форме мультиобъекта. Для этого следует выбрать отметку у свойства **Multiple instances** (Несколько экземпляров). Однако для объекта класса Устройство чтения карточки это свойство следует оставить пустым, поскольку данный объект присутствует в модели в единственном экземпляре.

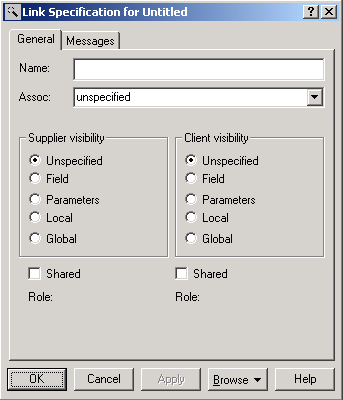
**2.Добавление связи и редактирование ее свойств**

Для добавления связи между предварительно размещенными на диаграмме объектами нужно с помощью левой кнопки мыши нажать кнопку с изображением связи на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении одного объекта на диаграмме и отпустить ее на изображении другого объекта. В результате этих действий на диаграмме появится изображение связи, например, соединяющей объект класса Клиент Банкомата (актера) с объектом класса Устройство чтения карточки (рис. 7.3). Поскольку кнопка с изображением актера отсутствует на специальной панели инструментов диаграммы кооперации, соответствующий объект следует предварительно поместить на диаграмму способом перетаскивания пиктограммы актера из браузера проекта.



**Рис. 7.3.** Диаграмма кооперации после добавления связи между объектом класса Клиент Банкомата (актером) и объектом класса Устройство чтения карточки

По умолчанию каждая добавляемая связь считается анонимной. При необходимости можно задать имя связи с помощью диалогового окна спецификации свойств данной связи (рис. 7.4).



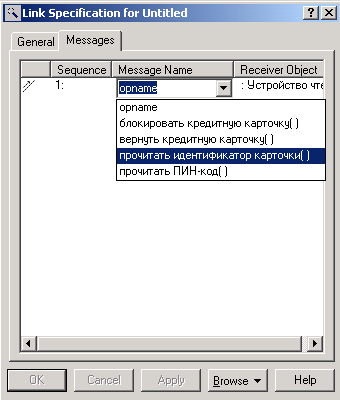
**Рис. 7.4.** Диалоговое окно редактирования свойств связи

Кроме имени связи можно также задать: имя ассоциации, видимость соответствующей пары объектов и наличие общих ролей. Однако более важной представляется следующая вкладка **Messages** ( сообщения ), служащая для спецификации сообщений, передаваемых между соответствующей парой объектов.

**3.Добавление сообщения и редактирование его свойств**

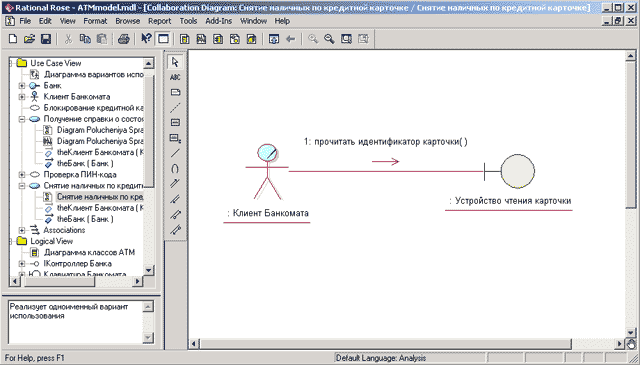
Добавить сообщения на диаграмму кооперации можно несколькими способами. Стандартный способ заключается в использовании кнопки с пиктограммой сообщения на специальной панели инструментов. В этом случае необходимо левой кнопкой мыши нажать кнопку с изображением прямого или обратного сообщения на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении линии связи на диаграмме и отпустить ее. В результате этих действий на диаграмме рядом с линией связи появится изображение стрелки сообщения.

Однако более удобным представляется способ добавления сообщений с помощью диалогового окна свойств связей. Для этого двойным щелчком на линии связи вызывается окно ее свойств и раскрывается вкладка **Messages** ( сообщения ). После этого следует выполнить операцию контекстного меню **Insert To** (Вставить в направлении), в результате чего появляется вложенный список с предложением выбрать одну из операций целевого класса для спецификации имени сообщения (рис.7.5).



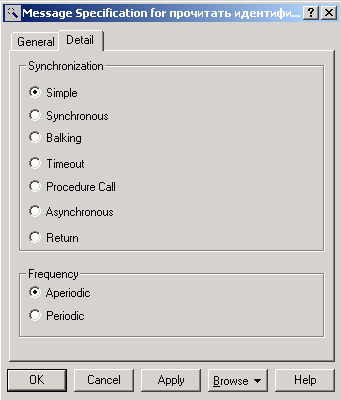
**Рис. 7.5.** Диалоговое окно добавления сообщения для выбранной связи

Для рассматриваемой модели банкомата для первого сообщения следует выбрать операцию прочитать идентификатор карточки(). После выбора операции для данного сообщения оно добавляется в список сообщений данной связи, а рядом с линией связи на диаграмме кооперации появится стрелка с номером и именем этого сообщения (рис. 7.6).



**Рис. 7.6.** Диаграмма кооперации после добавления связи между объектом класса Клиент Банкомата (актером) и объектом класса Устройство чтения карточки

Кроме имени сообщения можно также задать стереотип синхронизации и частоту передачи. Для этой цели следует воспользоваться диалоговым окном спецификации свойств сообщений (рис. 7.7), которое можно открыть двойным щелчком на имени сообщения в списке рассматриваемой вкладки **Messages** окна спецификации свойств связи.



**Рис. 7.7.** Диалоговое окно спецификации свойств сообщения

Группа свойств **Synchronization** (Синхронизация) предназначена для определения способа синхронизации передаваемого сообщения. При изменении этого свойства изменяется графическое изображение стрелки соответствующего сообщения. Характеристика отдельных свойств синхронизации сообщений и графическое изображение соответствующих стрелок сообщений приводится в следующей таблице (табл. 7.2).

Таблица 7.2. Характеристика свойств синхронизации сообщений

|  |  |  |
| --- | --- | --- |
| Название свойства | Графическое изображение стрелки | Назначение свойства |
| Simple (Простое) | Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_20.png | Данное сообщение выполняется в одном потоке управления. Это свойство задается добавляемому на диаграмму сообщению по умолчанию |
| Synchronous (Синхронное) | Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_21.png | После передачи данного сообщения клиент ожидает ответа от объекта-приемника о результате выполнения соответствующей операции |
| Balking (С отказом) | Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_22.png | После передачи данного сообщения объект-приемник отказывает клиенту в выполнении соответствующей операции, если он занят выполнением других операций |
| Timeout (С ожиданием) | Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_23.png | После передачи данного сообщения объект-приемник может поместить данное сообщение в очередь с ограниченным временем ожидания, если он занят выполнением других операций |
| Procedure Call (Вызов процедуры) | Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_24.png | Клиент посылает данное сообщение объекту-приемнику и, чтобы продолжить свою работу ожидает, пока вся дальнейшая вложенная последовательность сообщений не будет обработана приемником |
| Asynchronous (Асинхронное) | Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_25.png | Клиент посылает данное сообщение и продолжает свою работу, не ожидая подтверждения от объекта-приемника о получении этого сообщения. При этом соответствующая операция может быть как выполнена, так и не выполнена |
| Return (Возврат) | Описание: http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/7/files/7_26.png | Данное сообщение посылается клиенту после окончания выполнения вызова процедуры |

Группа свойств **Frequency** (Частота) предназначена для указания периодического характера передачи сообщения. При изменении этого свойства графическое изображение стрелки соответствующего сообщения не изменяется. Свойство **Aperiodic** (Апериодическое) означает, что сообщение посылается клиентом нерегулярно. При этом сообщение может быть отправлено один или несколько раз через различные промежутки времени. Это свойство задается для сообщения по умолчанию. Свойство **Periodic** (Периодическое) означает, что сообщение регулярно посылается клиентом через определенные промежутки времени.

Применительно для модели банкомата можно оставить рассмотренные свойства сообщений без изменения, в том виде, в каком они определены по умолчанию программой IBM Rational Rose .

**Диаграмма Развертывания**

Диаграмма развертывания является второй составной частью физического представления модели и разрабатывается, как правило, для территориально распределенных систем. Для разработки диаграмм компонентов в браузере проекта предназначено отдельное представление развертывания (**Deployment View**), в котором уже содержится диаграмма развертывания с пустым содержанием и без собственного имени.

***Технология работы:***

Активизация диаграммы развертывания может быть выполнена одним из следующих способов:

* Щелкнуть на кнопке с изображением диаграммы развертывания на стандартной панели инструментов.
* Дважды щелкнуть на пиктограмме представления развертывания ( **Deployment View** ) в браузере проекта.
* Выполнить операцию главного меню: **Browse \toDeployment Diagram** (Обзор \toДиаграмма развертывания).

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы развертывания и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы развертывания (табл. 7.3).

Таблица 7.3. Назначение кнопок специальной панели инструментов диаграммы развертывания

|  |  |  |
| --- | --- | --- |
| **Графическое изображение** | **Всплывающая подсказка** | **Назначение кнопки** |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/13/files/13_1.png | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/13/files/13_2.png | Text Box | Добавляет на диаграмму текстовую область |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/13/files/13_3.png | Note | Добавляет на диаграмму примечание |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/13/files/13_4.png | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/13/files/13_5.png | Processor | Добавляет на диаграмму процессор |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/13/files/13_6.png | Connection | Добавляет на диаграмму отношение соединения |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/13/files/13_7.png | Device | Добавляет на диаграмму устройство |

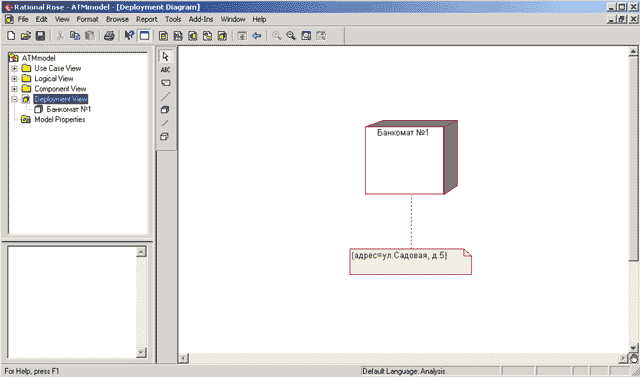
Как видно из этой таблицы, по умолчанию на панели инструментов присутствуют все графические элементы из рассмотренных ранее элементов диаграммы развертывания, поэтому изменять специальную панель нет необходимости. Работа с диаграммой развертывания состоит в создании процессоров и устройств, их спецификации, установлении связей между ними, а также добавлении и спецификации процессов.

**1.Добавление узла на диаграмму развертывания и редактирование его свойств**

Для добавления узла на диаграмму развертывания нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы требуемого узла (процессора или устройства) на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. Добавить процессор на диаграмму развертывания можно также с помощью операции главного меню: **Tools \toCreate \toProcessor** или с помощью операции контекстного меню: **New \toProcessor**, предварительно выделив представление развертывания в браузере проекта. Аналогично добавить устройство на диаграмму можно также с помощью операции главного меню: **Tools \toCreate \toDevice** или с помощью операции контекстного меню: **New \toDevice**, предварительно выделив представление развертывания в браузере проекта.

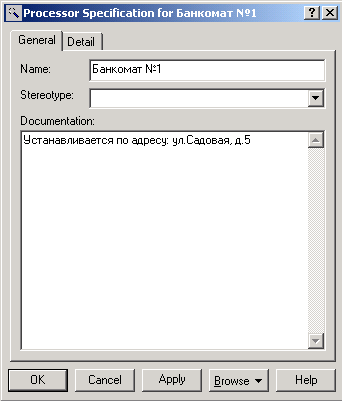
В результате этих действий на диаграмме развертывания появится изображение узла требуемого типа с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию, которое разработчику следует изменить. При этом следует иметь в виду, что в среде IBM Rational Rose под процессором понимается ресурсоемкий узел, а под устройством - нересурсоемкий узел.

Продолжая разработку модели системы управления банкоматом, построим для нее диаграмму развертывания. С этой целью в качестве первого узла выберем тип процессор и зададим ему имя Банкомат №1, для которого в форме примечания укажем помеченное значение: {адрес = ул. Садовая, д.5}. Это значение служит для спецификации конкретного адреса одного из банкоматов системы (рис. 7.8).



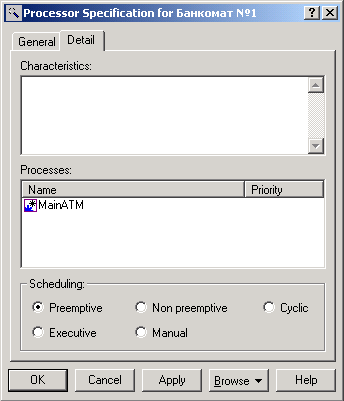
**Рис. 7.8.** Диаграмма развертывания после добавления узла Банкомат № 1

Для каждого процессора можно специфицировать различные свойства, такие как стереотип, характеристику, процессы и их приоритет. Спецификация этих свойств осуществляется с помощью диалогового окна спецификации свойств процессора (рис. 7.9).



**Рис. 7.9.** Диалоговое окно спецификации свойств узла Банкомат № 1

При этом на вкладке **General** (Общие) можно только изменить имя процессора, ввести текст стереотипа, предложенный самим разработчиком, и текст документации, поясняющий особенности физического размещения данного компонента. На вкладке **Detail** (Подробно) окна спецификации свойств процессора можно определить его характеристики, выбрать процессы и вариант планирования его работы (рис.7.9).



**Рис.7.9.** Диалоговое окно спецификации свойств узла Банкомат № 1, открытое на вкладке Detail (Подробно)

Характеристики процессора, такие как его быстродействие и объем оперативной памяти, могут быть записаны в форме текста в многостраничное поле с именем **Characteristics**. В поле **Processes** (Процессы) можно задать некоторый процесс, который предполагается реализовать на данном процессоре. С этой целью необходимо выполнить операцию контекстного меню **Insert** (Вставить) и ввести текст имени процесса. Далее можно задать приоритет процесса, введя некоторое число в соответствующее поле ввода.

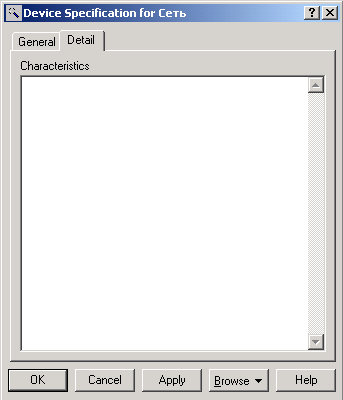
При наличии у процессора нескольких процессов может быть дополнительно определена процедура планирования их выполнения. Для спецификации процедуры планирования процессора могут быть использованы следующие варианты выбора в группе **Scheduling**:

* **Preemptive** (С приоритетом) - определяет процедуру планирования, при которой процесс с большим приоритетом будет иметь преимущество при использовании ресурсов процессора по сравнению с менее приоритетными процессами.
* **Non preemptive** (Без приоритета) - определяет процедуру планирования, при которой все приоритеты процессов игнорируются. При этом текущий процесс выполняется до своего завершения, после чего может быть начато выполнение следующего процесса.
* **Cyclic** (Циклический) - определяет процедуру планирования, при которой приоритеты процессов также игнорируются. Все процессы выполняются циклически по кругу, при этом каждому из них выделяется фиксированное время на выполнение, по прошествии которого управление передается следующему процессу.
* **Executive** (Исполнительный) - определяет процедуру планирования, для которой существует некоторый алгоритм, предназначенный для управления отдельными процессами.
* **Manual** (Вручную) - определяет процедуру планирования, при которой планирование выполнения процессов осуществляется пользователем.

Для отображения информации о процессах, выполняемых на отдельных процессорах, представленных на диаграмме развертывания, следует выполнить операцию контекстного меню **Show Processes** (Показать процессы). Для отображения информации о процедуре планирования отдельных процессов на выбранном процессоре следует выполнить операцию контекстного меню **Show Scheduling** (Показать планирование).

Продолжая разработку диаграммы развертывания для модели банкомата, следует добавить второй узел типа устройство ( **Device** ) с именем Сеть, для которого задать стереотип <<закрытая сеть>>. При этом для задания стереотипа следует ввести его текст без угловых кавычек в строку с именем **Stereotype**.

Для устройства набор редактируемых свойств меньше, поэтому для него с помощью соответствующего окна спецификации свойств можно определить: имя, стереотип, документацию и характеристику (рис.7.10). Этот факт согласуется с определением устройства как нересурсоемкого узла, на котором отсутствует процессор.

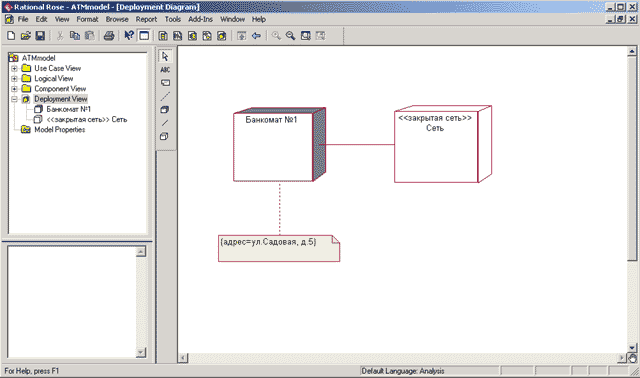


**Рис. 7.10.** Диалоговое окно спецификации свойств устройства Сеть, открытое на вкладке Detail (Подробно)

**2.Добавление соединения и редактирование его свойств**

Для добавления соединения между двумя узлами нужно с помощью левой кнопки мыши нажать кнопку с изображением соединения на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении одного из узлов на диаграмме и отпустить ее на изображении другого узла. Добавить соединения на диаграмму развертывания можно также с помощью операции главного меню: **Tools \toCreate \toConnection**.

В результате этих действий на диаграмме появится изображение соединения в форме линии без стрелок, соединяющей два выбранных узла. Применительно к диаграмме развертывания модели банкомата одним из рассмотренных способов следует добавить соединение для узлов с именами Банкомат №1 и Сеть (рис. 7.11).



**Рис. 7.11.** Диаграмма развертывания после добавления соединения между узлами Банкомат № 1 и Сеть

Для соединения набор редактируемых свойств аналогичен набору свойств устройства, поэтому для него с помощью соответствующего окна спецификации свойств можно определить только имя, стереотип, документацию и характеристику (рис. 7.10).

Следует отметить, что программа IBM Rational Rose не поддерживает возможности графического размещения внутри узлов развертываемых на них компонентов. Указать размещение компонентов модели в узлах диаграммы развертывания можно с помощью документации соответствующих узлов. Выполнить эти действия предлагается читателям самостоятельно в качестве упражнения. После построения диаграммы развертывания разработка визуальной модели системы управления банкоматом в нотации UML может считаться завершенной.

Дальнейшая работа с моделью зависит от целей выполнения проекта. Если проект не предполагает программную реализацию, то можно ограничиться формированием проектной документации. С этой целью следует выполнить операцию главного меню: **Report \toSoDA Report\_** (Отчет \toОтчет с помощью SoDA), в результате чего будет открыто диалоговое окно свойств для выбора шаблонов генерации отчета. После выбора шаблонов будет автоматически сгенерирован отчет о разрабатываемой модели в формате MS Word с использованием специального средства IBM Rational SoDA, если оно доступно в системе после инсталляции IBM Rational Rose .

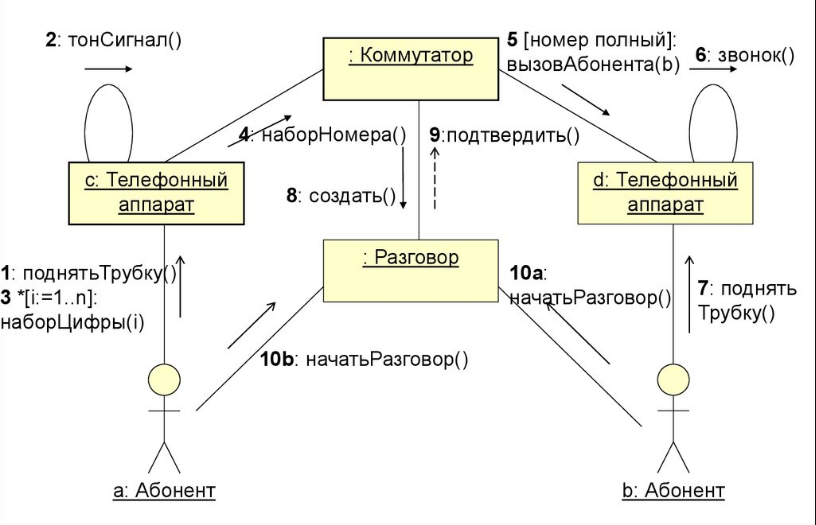


Рис. 7.12 – Пример диаграммы кооперации

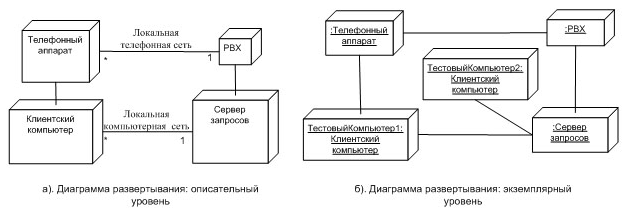


Рис. 7.13 – Пример диаграммы развертывания

**Задание.**

Изучить теоретические сведения по диаграммам Кооперации и Развертывания. Создать данные диаграммы по своей предметной области, ссылаясь на примеры диаграмм (рис.7.12-7.13).

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.

**Лабораторная работа №8. Построение диаграмм Деятельности, Состояния и Классов.**

**Цель работы**: научиться создавать диаграмму деятельности, состояния и классов.

**Теоретические сведения**

**Диаграмма деятельности** в среде IBM Rational Rose, также как и диаграмма состояний, может относиться к отдельному классу, операции класса, варианту использования, пакету или представлению.

***Технология работы:***

Для того чтобы построить диаграмму деятельности, ее вначале необходимо создать и активизировать.

Начать построение диаграммы деятельности для выбранного элемента модели или моделируемой системы в целом можно одним из следующих способов:

* Щелкнуть на кнопке с изображением диаграммы состояний на стандартной панели инструментов, после чего следует выбрать представление и тип разрабатываемой диаграммы - диаграмма деятельности.
* Выделить логическое представление ( **Logical View** ) или представление вариантов использования ( **Use Case View** ) в браузере проекта и выполнить операцию контекстного меню: **New \toActivity Diagram** (Новая \toДиаграмма деятельности ).
* Раскрыть логическое представление ( **Logical View** ) в браузере проекта и выделить рассматриваемый класс, операцию класса, пакет, или раскрыть представление вариантов использования ( **Use Case View** ) и выбрать вариант использования, после чего выполнить операцию контекстного меню: **New \toActivity Diagram** (Новая \toДиаграмма деятельности ).
* Выполнить операцию главного меню: **Browse \toState Machine Diagram** (Обзор \toДиаграмма состояний), после следует чего выбрать представление и тип разрабатываемой диаграммы - диаграмма деятельности.

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы деятельности и специальная панель инструментов, содержащая кнопки с изображением графических элементов, необходимых для разработки диаграммы деятельности (табл. 1). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 1. Назначение кнопок специальной панели инструментов диаграммы деятельности

|  |  |  |
| --- | --- | --- |
| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_1.png | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_2.png | Text Box | Добавляет на диаграмму текстовую область |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_3.png | Note | Добавляет на диаграмму примечание |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_4.png | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_5.png | State | Добавляет на диаграмму состояние |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_6.png | Activity | Добавляет на диаграмму деятельность |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_7.png | Start State | Добавляет на диаграмму начальное состояние |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_8.png | End State | Добавляет на диаграмму конечное состояние |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_9.png | State Transition | Добавляет на диаграмму переход |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_10.png | Transition to Self | Добавляет на диаграмму рефлексивный переход |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_11.png | Horizontal Synchronization | Добавляет на диаграмму горизонтально расположенный символ синхронизации |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_12.png | Vertical Synchronization | Добавляет на диаграмму вертикально расположенный символ синхронизации |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_13.png | Decision | Добавляет на диаграмму символ принятия решения для альтернативных переходов |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_14.png | Swimlane | Добавляет на диаграмму дорожку |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_15.png | Object | Добавляет на диаграмму объект (по умолчанию отсутствует) |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_16.png | Object Flow | Добавляет на диаграмму стрелку потока объектов (по умолчанию отсутствует) |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_17.jpg | Business Activity | Добавляет на диаграмму бизнес-деятельность (по умолчанию отсутствует) |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/10/files/10_18.jpg | Business Transaction | Добавляет на диаграмму бизнес-транзакцию (по умолчанию отсутствует) |

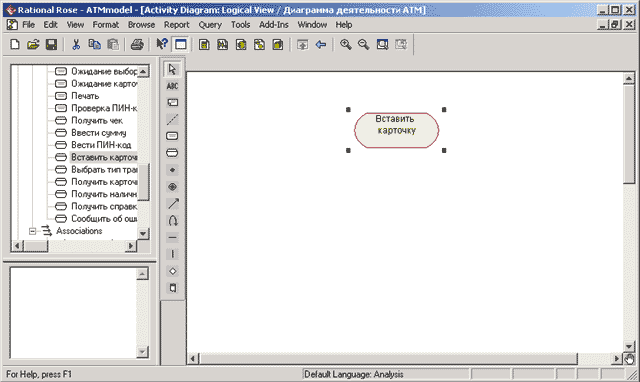
Как видно из этой таблицы, по умолчанию на панели инструментов отсутствуют некоторые графические элементы, а именно - кнопки с пиктограммами объекта и потока объектов. При необходимости их можно добавить на специальную панель диаграммы деятельности стандартным способом, который был описан ранее.

Для разрабатываемого проекта системы управления банкоматом диаграмма деятельности описывает последовательность действий клиента при использовании банкомата. Для удобства можно включить эту диаграмму в логическое представление, для чего необходимо в браузере проекта выделить логическое представление (**Logical View**) и выполнить операцию контекстного меню: **New \toActivity Diagram** (Новая \toДиаграмма деятельности ). Продолжая разработку проекта по моделированию системы управления банкоматом, можно приступить к разработке новой диаграммы деятельности. С этой целью для диаграммы деятельности модели банкомата зададим имя Диаграмма деятельности ATM, а в секцию ее документации введем текст "Диаграмма деятельности описывает последовательность действий клиента при использовании банкомата".

**1.Добавление деятельности на диаграмму деятельности и редактирование ее свойств**

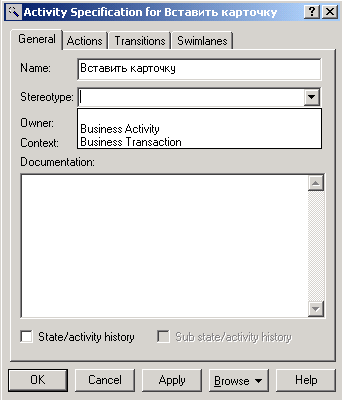
Для добавления деятельности на диаграмму деятельности нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы деятельности на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение деятельности с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию, которое разработчику следует изменить. Добавить деятельность на диаграмму можно также с помощью операции главного меню: **Tools \toCreate \toActivity** или с помощью операции контекстного меню: **New \toActivity**, предварительно выделив диаграмму деятельности в браузере проекта.

В результате этих действий на диаграмме появится изображение деятельности с именем **NewActivity**, предложенное программой по умолчанию. Начиная построение диаграммы деятельности модели банкомата, для первой добавленной деятельности зададим имя Вставить карточку (рис. 1).



**Рис. 1.** Диаграмма деятельности после добавления на нее деятельности Вставить карточку

После добавления деятельности на диаграмму деятельности можно открыть диалоговое окно спецификации ее свойств и определить дополнительные свойства деятельности, доступные на соответствующих вкладках (рис. 2).



**Рис. 2.** Диалоговое окно спецификации свойств деятельности

При этом для деятельности становятся доступными для выбора два стереотипа: **Business Activity** (Бизнес-деятельность) и **Business Transaction** (Бизнес-транзакция), которые имеют собственное графическое изображение (см. табл. 10.1). На вкладке **Transitions** ( Переходы ) окна спецификации свойств деятельности можно определять и редактировать переходы, которые входят и выходят из рассматриваемой деятельности. Последняя вкладка **Swimlanes** (Дорожки) служит для спецификации дорожки, на которую помещается рассматриваемая деятельность.

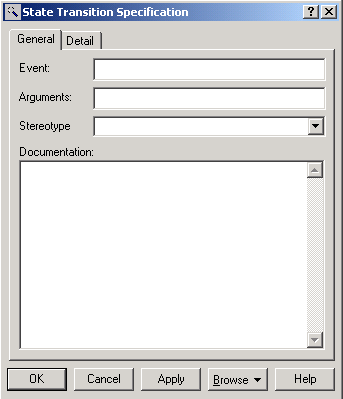
Хотя программа IBM Rational Rose позволяет определить свойства деятельности, доступные на вкладке **Actions** (Действия), следует помнить, что внутренние действия являются свойствами общего понятия состояния, а внутренняя деятельность служит именем собственно деятельности, помещаемой на диаграмму деятельности. Поэтому для деятельности во избежание недоразумений лучше оставить эту вкладку пустой.

**2.Добавление перехода и редактирование его свойств**

Добавление перехода на диаграмму деятельности полностью аналогично диаграмме состояний. А именно, для добавления перехода между двумя деятельностями нужно с помощью левой кнопки мыши нажать кнопку с изображением перехода на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении исходной деятельности на диаграмме и отпустить ее на изображении целевой деятельности. В результате этих действий на диаграмме появится изображение перехода, соединяющего две выбранных деятельности. Если в качестве одной из деятельностей является символ ветвления или соединения, то порядок добавления перехода сохраняется прежним.

Следует заметить, что при наличии в проекте законченной диаграммы состояний попытка добавить начальное состояние на диаграмму деятельности с помощью кнопки специальной панели инструментов окажется безуспешной. В этом случае программа IBM Rational Rose фиксирует наличие в модели начального состояния и не позволит добавить его с помощью соответствующей кнопки на разрабатываемые диаграммы состояний или деятельности. Решить данную проблему можно посредством перетаскивания с помощью мыши начального состояния из браузера проекта на любую из вновь разрабатываемых диаграмм.

После добавления перехода на диаграмму деятельности становятся доступными для редактирования его свойства в специальном диалоговом окне (рис. 3), которое можно открыть по двойному щелчку левой кнопкой мыши на изображении перехода.



**Рис. 3.** Диалоговое окно спецификации свойств перехода

При спецификации свойств переходов следует помнить, что все переходы на диаграмме деятельности является нетриггерными, т.е. не имеют имен событий. По этой причине поле ввода с именем **Event** (Событие) для всех переходов должно оставаться пустым. Но все переходы, выходящие из символов ветвления (решения), должны иметь сторожевые условия, которые специфицируются на вкладке **Detail** (Подробно) диалогового окна спецификации свойств перехода.

**Диаграмма состояний**

В среде IBM Rational Rose этот тип диаграмм может относиться к отдельному классу, операции класса, варианту использования, пакету или представлению. Для того чтобы построить диаграмму состояний, ее вначале необходимо создать и активизировать.

Начать построение диаграммы состояний для выбранного элемента модели или моделируемой системы в целом можно одним из следующих способов:

* Щелкнуть на кнопке с изображением диаграммы состояний на стандартной панели инструментов, после чего следует выбрать представление и тип разрабатываемой диаграммы - новая диаграмма состояний.
* Выделить логическое представление ( **Logical View** ) или представление вариантов использования ( **Use Case View** ) в браузере проекта и выполнить операцию контекстного меню: **New \toStatechart Diagram** (Новая \toДиаграмма состояний ).
* Раскрыть логическое представление ( **Logical View** ) в браузере проекта и выделить рассматриваемый класс, операцию класса, пакет, или раскрыть представление вариантов использования ( **Use Case View** ) и выбрать вариант использования, после чего выполнить операцию контекстного меню: **New \toStatechart Diagram** (Новая \toДиаграмма состояний ).
* Выполнить операцию главного меню: **Browse \toState Machine Diagram** (Обзор \toДиаграмма состояний ), после чего следует выбрать представление и тип разрабатываемой диаграммы.

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы состояний и специальная панель инструментов, содержащая кнопки с изображением графических элементов модели, необходимых для разработки диаграммы состояний (табл. 2). Назначение отдельных кнопок панели можно узнать из всплывающих подсказок.

Таблица 2. Назначение кнопок специальной панели инструментов диаграммы состояний

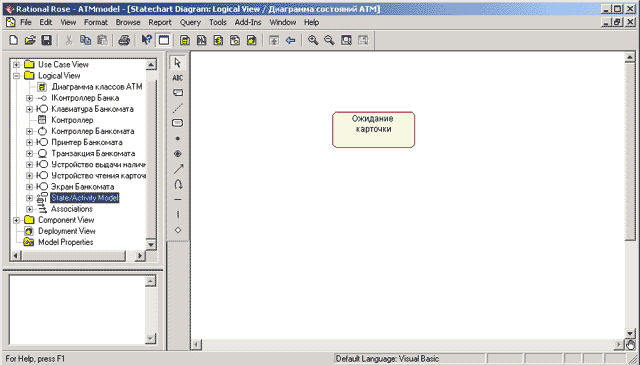
|  |  |  |
| --- | --- | --- |
| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_1.png | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_2.png | Text Box | Добавляет на диаграмму текстовую область |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_3.png | Note | Добавляет на диаграмму примечание |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_4.png | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_5.png | State | Добавляет на диаграмму состояние |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_6.png | Start State | Добавляет на диаграмму начальное состояние |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_7.png | End State | Добавляет на диаграмму конечное состояние |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_8.png | State Transition | Добавляет на диаграмму переход |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_9.png | Transition to Self | Добавляет на диаграмму рефлексивный переход |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_10.png | Horizontal Synchronization | Добавляет на диаграмму горизонтально расположенный символ синхронизации (по умолчанию отсутствует) |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_11.png | Vertical Synchronization | Добавляет на диаграмму вертикально расположенный символ синхронизации (по умолчанию отсутствует) |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/9/files/9_12.png | Decision | Добавляет на диаграмму символ принятия решения для альтернативных переходов (по умолчанию отсутствует) |

По умолчанию на специальной панели инструментов могут отсутствовать кнопки с тремя последними графическими элементами из таблицы 1. При необходимости их можно добавить на специальную панель диаграммы состояний. Продолжая разработку проекта по моделированию системы управления банкоматом, можно приступить к разработке новой диаграммы состояний. С этой целью для диаграммы состояний модели банкомата зададим имя Диаграмма состояний ATM, а в секцию ее документации введем текст "Диаграмма состояний описывает конечный автомат банкомата".

**1.Добавление состояния на диаграмму состояний и редактирование его свойств**

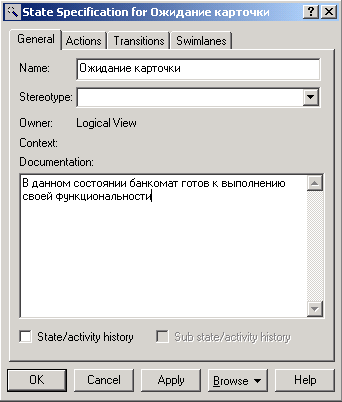
Для добавления состояния на диаграмму состояний необходимо с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы состояния на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение состояния с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию, которое разработчику следует изменить.

Для диаграммы состояний модели банкомата в качестве имени первого добавленного состояния изменим предложенное программой по умолчанию имя NewState на Ожидание карточки (рис. 4). Задать имя состояния можно либо непосредственно при добавлении нового состояния на диаграмму состояний, либо открыв окно спецификации свойств нового состояния.



**Рис. 4.** Диаграмма состояний после добавления на нее состояния Ожидание карточки

Для добавленного состояния можно открыть диалоговое окно его свойств двойным щелчком левой кнопкой мыши на изображении этого элемента на диаграмме. В этом случае активизируется диалоговое окно со специальными вкладками, в поля которых можно занести всю информацию по данному состоянию (рис. 5).



**Рис. 5.** Диалоговое окно спецификации свойств состояния

При необходимости в диалоговом окне спецификации свойств выбранного состояния можно задать вложенное историческое состояние. Для этого следует выставить отметку у свойства **State/activity history** ( Историческое состояние /деятельность) и нажать кнопку **Apply**. В результате внутри исходного состояния появится вложенное историческое состояние (рис. 5, а).



**Рис. 5.** Добавление вложенного исторического состояния (а) и состояния глубокой истории (б) для состояния Ожидание карточки

Чтобы обычное историческое состояние превратить в состояние глубокой истории, следует дополнительно выставить отметку у свойства **Sub state/activity history** (Историческое под-состояние/деятельность), которое становится доступным для редактирования после выбора первого свойства, и нажать кнопку **Apply**. В результате внутри исходного состояния появится вложенное состояние глубокой истории (рис. 5, б).

Чтобы обычное состояние превратить в композит, следует при добавлении нового состояния поместить его внутри границы того состояния, которое необходимо сделать композитным. В результате внутри исходного состояния появится новое вложенное состояние с именем **NewState**, которое при перемещении композита в области диаграммы состояний всегда будет находиться внутри своего композита (рис. 6).



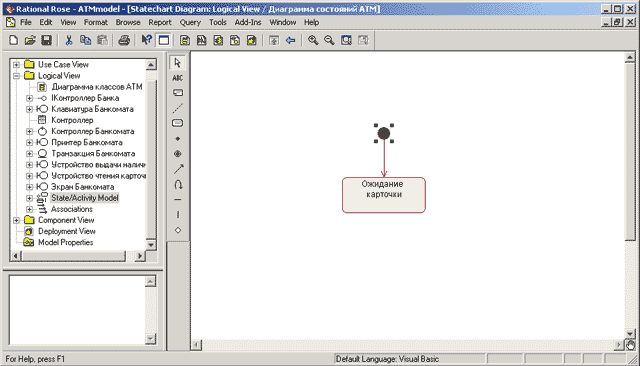
**Рис. 6.** Превращение состояния Ожидание карточки в композитное состояние

Рассмотренные выше действия приведены только с целью иллюстрации особенностей спецификации исторических и вложенных подсостояний и не относятся к разрабатываемой модели банкомата.

Дополнительно можно определить следующие свойства состояний: задать текстовый стереотип состояния, определить внутренние действия на входе и выходе, а также внутреннюю деятельность. Эти свойства доступны для редактирования на вкладке **General** (Общие) и **Actions** (Действия). На вкладке **Transitions** (Переходы) можно определять и редактировать переходы, которые входят и выходят из рассматриваемого состояния. Последняя вкладка **Swimlanes** (Дорожки) служит для спецификации дорожек, которые, в контексте языка UML, определяются для диаграммы деятельности.

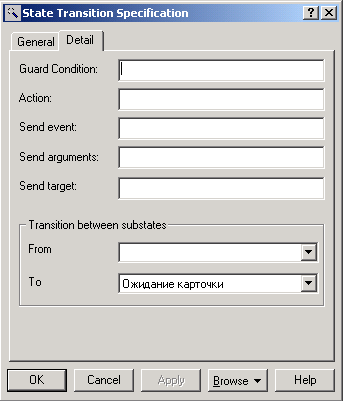
**2.Добавление перехода и редактирование его свойств**

Для добавления перехода между двумя состояниями нужно с помощью левой кнопки мыши нажать кнопку с изображением перехода на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении исходного состояния на диаграмме и отпустить ее на изображении целевого состояния. В результате этих действий на диаграмме появится изображение перехода, соединяющего два выбранных состояния. Продолжая разработку модели системы управления банкоматом, добавим на диаграмму состояний начальное состояние ( **Start State** ) и соединим его переходом с состоянием Ожидание карточки (рис. 7).



**Рис.7.** Диаграмма состояний после добавления на нее перехода из начального состояния в состояние Ожидание карточки

После добавления перехода на диаграмму состояний можно открыть диалоговое окно его свойств и специфицировать дополнительные свойства, доступные на соответствующих вкладках (рис. 8). Следует обратить внимание на две первые строки вкладки **Detail** (Подробно), которые представляются наиболее важными из свойств перехода. Первое поле ввода **Guard Condition** служит для задания сторожевого условия, которое определяет правило срабатывания соответствующего перехода. Во втором поле ввода **Action** можно специфицировать действие, которое происходит при срабатывании перехода до того, как моделируемая система попадет в целевое состояние.



**Рис.8.** Диалоговое окно спецификации свойств перехода, открытое на вкладке Detail (Подробно)

При необходимости можно определить сообщение о событии, происходящем при срабатывании перехода, а также визуализировать вложенность состояний и подключить историю отдельных состояний.

Диаграмма классов является основным логическим представлением модели и содержит детальную информацию о внутреннем устройстве объектно-ориентированной программной системы или, используя современную терминологию, об архитектуре программной системы.

**Диаграмма классов** определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Диаграмма классов UML - это граф, узлами которого являются элементы статической структуры проекта (классы, интерфейсы), а дугами - отношения между узлами (ассоциации, наследование, зависимости).

На диаграмме классов изображаются следующие элементы:

* Пакет (package) - набор элементов модели, логически связанных между собой;
* Класс (class) - описание общих свойств группы сходных объектов;
* Интерфейс (interface) - абстрактный класс, задающий набор операций, которые объект произвольного класса, связанного с данным интерфейсом, предоставляет другим объектам.

*Класс* - это группа сущностей (объектов), обладающих сходными свойствами, а именно, данными и поведением. Отдельный представитель некоторого класса называется объектом класса или просто объектом.

Под поведением объекта в UML понимаются любые правила взаимодействия объекта с внешним миром и с данными самого объекта.

На диаграммах класс изображается в виде прямоугольника со сплошной границей, разделенного горизонтальными линиями на 3 секции:

* Верхняя секция (секция имени) содержит имя класса и другие общие свойства (в частности, стереотип).
* В средней секции содержится список атрибутов
* В нижней - список операций класса, отражающих его поведение (действия, выполняемые классом).

Любая из секций атрибутов и операций может не изображаться (а также обе сразу). Для отсутствующей секции не нужно рисовать разделительную линию и как-либо указывать на наличие или отсутствие элементов в ней.

*Стереотипы классов* – это механизм, позволяющий разделять классы на категории.

В языке UML определены три основных стереотипа классов:

* Boundary (граница);
* Entity (сущность);
* Control (управление).

*Граничными классами* (boundary classes) называются такие классы, которые расположены на границе системы и всей окружающей среды. Это экранные формы, отчеты, интерфейсы с аппаратурой (такой как принтеры или сканеры) и интерфейсы с другими системами.

Чтобы найти граничные классы, надо исследовать диаграммы вариантов использования. Каждому взаимодействию между действующим лицом и вариантом использования должен соответствовать, по крайней мере, один граничный класс. Именно такой класс позволяет действующему лицу взаимодействовать с системой.

*Классы-сущности* (entity classes) содержат хранимую информацию. Они имеют наибольшее значение для пользователя, и потому в их названиях часто используют термины из предметной области. Обычно для каждого класса-сущности создают таблицу в базе данных.

*Управляющие классы* (control classes) отвечают за координацию действий других классов. Обычно у каждого варианта использования имеется один управляющий класс, контролирующий последовательность событий этого варианта использования. Управляющий класс отвечает за координацию, но сам не несет в себе никакой функциональности, так как остальные классы не посылают ему большого количества сообщений. Вместо этого он сам посылает множество сообщений. Управляющий класс просто делегирует ответственность другим классам, по этой причине его часто называют классом-менеджером.

Помимо упомянутых выше стереотипов можно создавать и свои собственные.

**Технологияработы**

Активизировать рабочее окно диаграммы классов можно несколькими способами:

* окно диаграммы классов появляется по умолчанию в рабочем окне диаграммы после создания нового проекта;
* щелкнуть на кнопке с изображением диаграммы классов на стандартной панели инструментов;
* раскрыть логическое представление (Logical View) в браузере проекта и дважды щелкнуть на пиктограмме **Main** (Главная);
* выполнить операцию главного меню: **Browse>Class Diagram** (Обзор>Диаграмма классов).

При этом появляется новое окно с чистым рабочим листом диаграммы классов и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы классов (табл. 3). Назначение отдельных кнопок панели можно узнать также из всплывающих подсказок.

Таблица 3. Назначение кнопок специальной панели инструментов для диаграммы классов

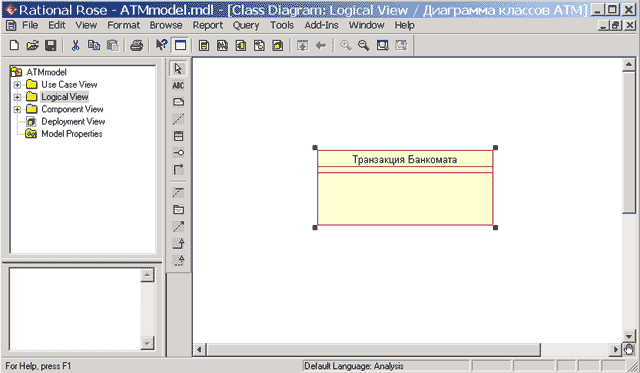
|  |  |  |
| --- | --- | --- |
| Графическое изображение | Всплывающая подсказка | Назначение кнопки |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_1.png | Selection Tool | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_2.png | Text Box | Добавляет на диаграмму текстовую область |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_3.png | Note | Добавляет на диаграмму примечание |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_4.png | Anchor Note to Item | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_5.png | Class | Добавляет на диаграмму класс |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_6.png | Interface | Добавляет на диаграмму интерфейс |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_7.png | Unidirectional Association | Добавляет на диаграмму направленную ассоциацию |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_8.png | Association Class | Добавляет на диаграмму ассоциацию класс |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_9.png | Package | Добавляет на диаграмму пакет |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_10.png | Dependency or Instantiates | Добавляет на диаграмму отношение зависимости |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_11.png | Generalization | Добавляет на диаграмму отношение обобщения |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/4/files/4_12.png | Realize | Добавляет на диаграмму отношение реализации |

На специальной панели инструментов по умолчанию присутствует только часть пиктограмм элементов, которые могут быть использованы для построения диаграммы классов. Добавить кнопки с пиктограммами других графических элементов таких как, например, отношения агрегации и композиции, шаблон, класс бизнес-сущность, управляющий класс, или удалить ненужные кнопки можно с помощью настройки специальной панели инструментов. Соответствующее диалоговое окно настройки специальной панели инструментов для диаграммы классов можно вызвать аналогично другим панелям с помощью операции контекстного меню **Customize** (Настройка) при позиционировании курсора на специальной панели инструментов.

**1.Добавление класса на диаграмму классов и редактирование его свойств**

Для добавления класса на диаграмму классов нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы класса на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. На диаграмме появится изображение класса с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию NewClass.

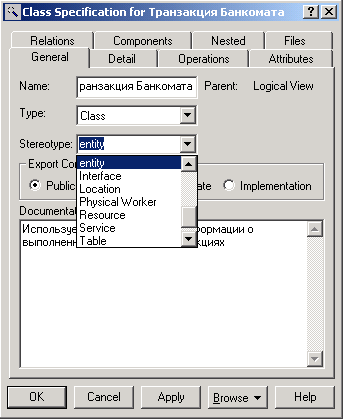
Продолжая разработку модели банкомата в качестве сквозного примера проекта, построим для этой модели следующую каноническую диаграмму - диаграмму классов. С этой целью следует изменить предложенное по умолчанию имя диаграммы Main на Диаграмма классов ATM, а имя добавленного на диаграмму класса - на Транзакция Банкомата (рис. 9).



**Рис. 9.** Диаграмма классов модели банкомата после добавления на нее класса Транзакция Банкомата

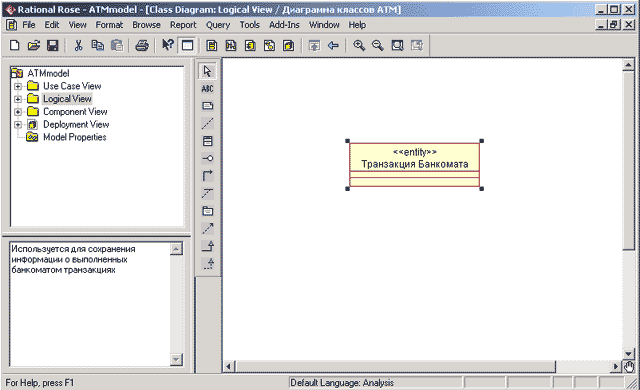
Поскольку разрабатываемая модель банкомата на начальных этапах работы над проектом используется для анализа общей архитектуры проекта и согласования ее с различными участниками рабочей группы, имена классов, их атрибутов и операций для большей наглядности, и понимания задаются на русском языке с пробелами и записываются символами кириллицы. В последующем по мере выполнения проекта и реализации модели на некотором языке программирования, имена соответствующих классов, атрибутов и операций должны быть преобразованы в символы латиницы. При этом имена этих элементов модели должны быть записаны без пробелов. В контексте управляемой моделью архитектуры первую модель еще называют независимой от платформы реализации, а вторую - зависимой от платформы реализации.

Для класса Транзакция Банкомата можно уточнить его назначение в модели с помощью указания стереотипа и пояснительного текста в форме документации. С этой целью двойным щелчком левой кнопкой мыши на изображении этого класса на диаграмме или в браузере проекта следует открыть диалоговое окно спецификации свойств этого класса (рис. 10) и на вкладке **General** (Общие) выбрать из вложенного списка **Stereotype** стереотип **entity** (сущность).



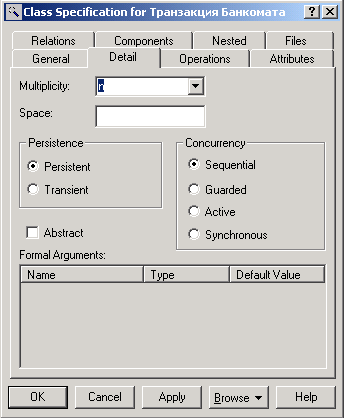
**Рис. 10.** Диалоговое окно спецификации свойств класса Транзакция Банкомата при выборе из вложенного списка стереотипа entity

Выбор данного стереотипа означает, что соответствующий класс предназначен для хранения информации, которая должна сохраняться в системе после уничтожения объектов данного класса. Далее в секцию документации данного класса можно ввести поясняющий текст: "Используется для сохранения информации о выполненных банкоматом транзакциях" и нажать кнопку **Apply** или **OK**, чтобы сохранить результаты редактирования свойств выбранного класса. После назначения стереотипа классу Транзакция банкомата текст данного стереотипа в угловых скобках появится выше имени данного класса (рис. 11).



**Рис. 11.** Диаграмма классов модели банкомата после выбора стереотипа для класса Транзакция Банкомата

Для отдельного класса можно уточнить также и другие его свойства, доступные для редактирования на вкладке **Detail** (Подробно) окна спецификации свойств этого класса. Например, на этой вкладке с помощью вложенного списка **Multiplicity** (Кратность) можно задать количество объектов или экземпляров данного класса, для чего следует выбрать строку с буквой n. Данное значение означает, что у класса Транзакция банкомата может быть любое конечное число экземпляров (рис. 12). Поле ввода с именем **Space** (Пространство) служит для указания объема абсолютной или относительной памяти, которая требуется, по оценке разработчика, для реализации каждого объекта данного класса. Применительно к рассматриваемой модели — это поле можно оставить пустым.



**Рис. 12.** Диалоговое окно спецификации свойств класса Транзакция Банкомата, открытое на вкладке Detail (Подробно)

Далее можно задать устойчивость классов в группе выбора **Persistence**. При этом выбор свойства **Persistent** (Устойчивый) означает, что информация об объектах данного класса должна быть сохранена в системе. Выбор свойства **Transient** (Временный) означает, что нет необходимости сохранять информацию об объектах данного класса в системе после завершения работы программного приложения. Применительно к рассматриваемой модели следует выбрать свойство **Persistent**.

В группе выбора **Concurrency** (Параллельность) можно специфицировать условия на возможность реализации объектов данного класса в параллельных потоках управления. Для выбора могут быть использованы следующие свойства:

* **Sequential** (Последовательный) - свойство по умолчанию, которое означает, что объекты класса будут вести себя нормально только при наличии одного потока управления, т. е. соответствующие операции объектов должны выполняться последовательно. В то же время при наличии нескольких потоков управления стабильное поведение объектов класса не гарантируется.
* **Guarded** (Безопасный) - означает, что при наличии нескольких потоков управления объекты класса будут вести себя ожидаемым от них образом. Для этого объекты в различных потоках должны взаимодействовать друг с другом для того, чтобы гарантировать отсутствие конфликта между ними.
* **Active** (Активный) - означает, что класс должен иметь свой собственный поток управления.
* **Synchronous** (Синхронный) - означает, что объекты класса будут вести себя ожидаемым от них образом при наличии нескольких потоков управления. При этом нет необходимости во взаимодействии объектов в различных потоках управления, поскольку объекты данного класса могут самостоятельно разрешать возможные конфликты.

Для того, чтобы специфицировать класс как абстрактный, т.е. не имеющий экземпляров, следует на этой же вкладке выставить отметку в свойстве **Abstract** (Абстрактный). Применительно к рассматриваемой модели для класса Транзакция банкомата следует выбрать свойства **Persistent** и **Sequential**, а отметку для свойства **Abstract** оставить пустой.

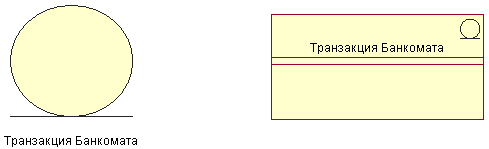
Следует заметить, что для предотвращения потери информации о разрабатываемой модели и результатов редактирования свойств ее графических элементов необходимо периодически сохранять модель во внешнем файле. Для этого следует выполнить операцию главного меню: **File>Save** (Файл>Сохранить) или нажать комбинацию клавиш: **Ctrl+S**. Для этой же цели служит соответствующая кнопка на стандартной панели инструментов (см. табл. 1.1).

**2.Стереотипы классов и их графическое представление**

На разрабатываемой диаграмме классов выбран текстовый способ изображения стереотипов классов, при котором стереотип записывается в угловых кавычках выше имени соответствующего класса. Программа IBM Rational Rose позволяет альтернативно представлять стереотипы в форме специальных графических изображений (как в браузере проекта) или в форме небольших декоративных значков в верхней секции прямоугольника класса на диаграмме, а также вообще отказаться от изображения стереотипов.

Изменить изображение стереотипа для отдельного класса можно, например, с помощью одной из вложенных операций контекстного меню: **Options>Stereotype Display** (Параметры>Изображение стереотипа). В качестве примера можно представить изображение класса Транзакция Банкомата в форме специальной графической пиктограммы стереотипа. С этой целью следует выполнить операцию контекстного меню: **Options>Stereotype Display>Icon** (Параметры>Изображение стереотипа>Пиктограмма). Соответствующее графическое изображение стереотипа <<entity>> для класса Транзакция Банкомата в форме пиктограммы будет иметь следующий вид (рис. 13).

Для сравнения можно выбрать изображение класса Транзакция Банкомата в форме декоративного графического стереотипа. С этой целью выполним операцию контекстного меню: **Options>Stereotype Display>Decoration** (Параметры>Изображение стереотипа>Декорация). Соответствующее графическое изображение стереотипа <<entity>> для класса Транзакция Банкомата в форме декорации будет иметь следующий вид (рис. 13).

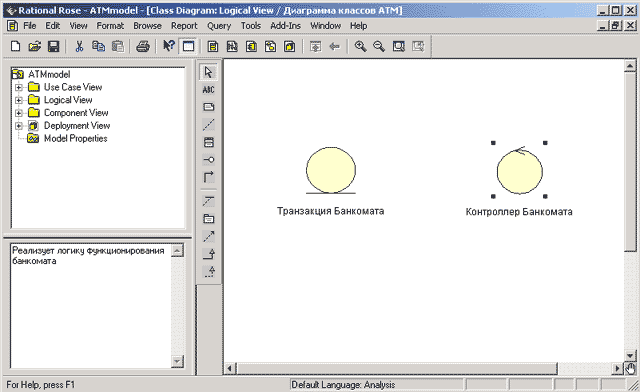


**Рис. 13.** Графические способы изображения стереотипа <<entity>> для класса Транзакция Банкомата

Изменить изображение стереотипов одновременно для нескольких классов диаграммы можно с помощью одной из вложенных операций главного меню: **Format>Stereotype Display** (Формат>Изображение стереотипов). В этом случае необходимо выделить все классы модели в окне диаграммы классов или в браузере проекта. Для выделения группы классов на диаграмме или в браузере проекта следует, удерживая нажатой клавишу **Ctrl** или **Shift** на клавиатуре, последовательно щелкать на их изображении левой кнопкой мыши.

Выделить все графические элементы на диаграмме классов, также как и на любой другой диаграмме модели, можно с помощью выполнения операции главного меню: **Edit>Select All** (Редактирование>Выделить все) или с помощью комбинации клавиш **Ctrl+А**. Следует отметить, что выбор того или иного способа изображения стереотипов классов на диаграмме классов определяется разработчиком исходя из его личных предпочтений, и не оказывает влияния на содержательный аспект логического представления модели.

Продолжая разработку модели банкомата, добавим на диаграмму второй класс с именем Контроллер Банкомата, для которого в окне спецификации свойств выберем стереотип **control** (управляющий класс), а в качестве документации введем текст: "Реализует логику функционирования банкомата". При этом атрибуты и операции у данного класса будут отсутствовать. Соответствующий фрагмент диаграммы классов после добавления управляющего класса Контроллер Банкомата будет иметь следующий вид (рис. 14).



**Рис. 14.** Фрагмент диаграммы классов модели банкомата после добавления на нее класса Контролер Банкомата

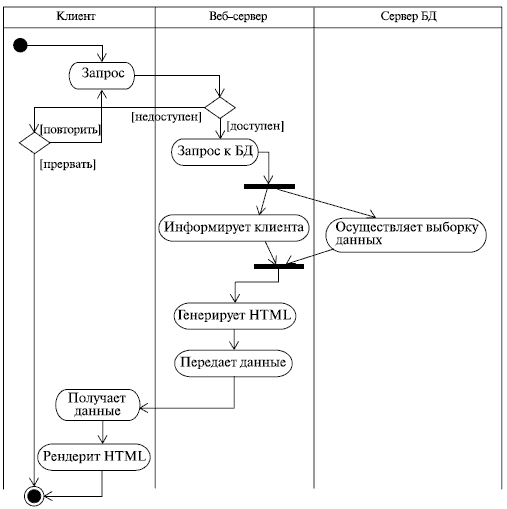


Рис. 15 – Диаграмма деятельности

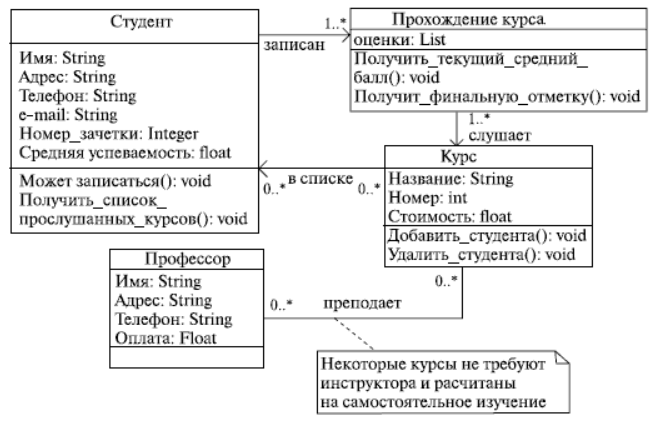


Рис. 16 – Диаграмма классов

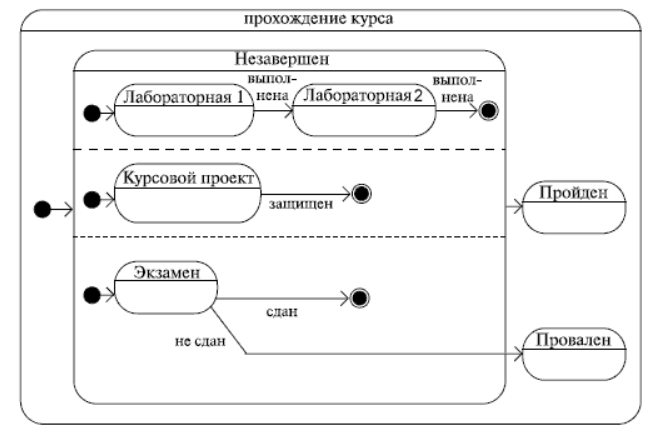


Рис. 17 – Диаграмма состояний

**Задание.**

Изучить теоретические сведения по диаграммам деятельности, состояний и классов. Создать данные диаграммы по своей предметной области, ссылаясь на примеры диаграмм (рис.15-17).

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.

**Лабораторная работа №9. Построение диаграммы Компонентов**

**Цель работы**: научиться создавать диаграмму Компонентов.

**Теоретические сведения**

**Диаграмма компонентов** служит частью физического представления модели, играет важную роль в процессе ООАП и является необходимой для генерации программного кода.

**Технология работы:**

Для разработки диаграмм компонентов в браузере проекта предназначено отдельное представление компонентов (**Component View**), в котором уже содержится диаграмма компонентов с пустым содержанием и именем по умолчанию **Main** (Главная).

Активизация диаграммы компонентов может быть выполнена одним из следующих способов:

* Щелкнуть на кнопке с изображением диаграммы компонентов на стандартной панели инструментов.
* Раскрыть представление компонентов в браузере (**Component View**) и дважды щелкнуть на пиктограмме **Main** (Главная).
* Через пункт меню **Browse \toComponent Diagram** (Браузер \toДиаграмма компонентов).

В результате выполнения этих действий появляется новое окно с чистым рабочим листом диаграммы компонентов и специальная панель инструментов, содержащая кнопки с изображением графических примитивов, необходимых для разработки диаграммы компонентов (табл. 9.1).

Таблица 9.1. Назначение кнопок специальной панели инструментов диаграммы компонентов

|  |  |  |  |
| --- | --- | --- | --- |
| Графическое изображение | Всплывающая подсказка | Назначение кнопки | |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_1.png | Selection Tool | | Превращает изображение курсора в форму стрелки для последующего выделения элементов на диаграмме |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_2.png | Text Box | | Добавляет на диаграмму текстовую область |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_3.png | Note | | Добавляет на диаграмму примечание |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_4.png | Anchor Note to Item | | Добавляет на диаграмму связь примечания с соответствующим графическим элементом диаграммы |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_5.png | Component | | Добавляет на диаграмму компонент |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_6.png | Package | | Добавляет на диаграмму пакет |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_7.png | Dependency | | Добавляет на диаграмму отношение зависимости |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_8.png | Subprogram Specification | | Добавляет на диаграмму спецификацию подпрограммы |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_9.png | Subprogram Body | | Добавляет на диаграмму тело подпрограммы |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_10.png | Main Program | | Добавляет на диаграмму главную программу |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_11.png | Package Specification | | Добавляет на диаграмму спецификацию пакета |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_12.png | Package Body | | Добавляет на диаграмму тело пакета |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_13.png | Task Specification | | Добавляет на диаграмму спецификацию задачи |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_14.png | Task Body | | Добавляет на диаграмму тело задачи |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_15.png | Generic Subprogram | | Добавляет на диаграмму типовую подпрограммы(по умолчанию отсутствует) |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_16.png | Generic Package | | Добавляет на диаграмму типовой пакет (по умолчанию отсутствует) |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_17.png | Database | | Добавляет на диаграмму базу данных (по умолчанию отсутствует) |

Как видно из этой таблицы, по умолчанию на панели инструментов отсутствуют только три графических элемента из рассмотренных ранее элементов диаграммы компонентов, а именно - кнопки с пиктограммами типовой подпрограммы, типового пакета и базы данных. При необходимости их можно добавить на специальную панель диаграммы компонента стандартным способом.

Графическое изображение этих стереотипов и их краткая характеристика приводятся в следующей таблице (табл. 9.2). При этом каждому из компонентов, как правило, соответствует отдельный файл исходной сборки программного приложения.

Таблица 2. Графическое изображение стереотипов компонентов и их характеристика

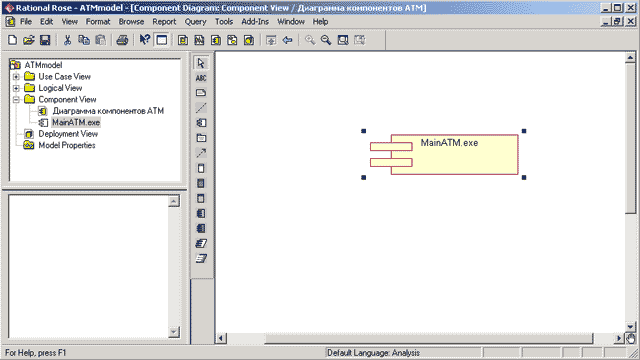
|  |  |  |
| --- | --- | --- |
| Графическое изображение и имя по умолчанию | Название стереотипа | Характеристика стереотипа компонента |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_18.png | Subprogram Specification | Спецификация подпрограммы. Содержит описание переменных, процедур и функций и не содержит определений классов |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_19.png | Subprogram Body | Тело подпрограммы. Содержит реализацию процедур и функций, не относящихся к каким-то классам, при этом не содержит определений классов или реализаций операций других классов |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_20.png | Main Program | Главная программа. Реализует базовую логику работы программного приложения и содержит ссылки на другие компоненты модели |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_21.png | Package Specification | Спецификация пакета. Содержит определение класса, его атрибутов и операций. В языке программирования С++ спецификации пакета соответствует отдельный файл с расширением "h" |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_22.png | Package Body | Тело пакета. Содержит код реализации операций класса. В языке программирования С++ спецификации пакета соответствует отдельный файл с расширением "cpp" |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_23.png | Task Specification | Спецификация задачи. Может содержать определение класса, его атрибутов и операций, которые предполагается использовать в независимом потоке управления |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_24.png | Task Body | Тело задачи. Может содержать реализацию операций класса, которые имеют независимый поток управления. |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_25.png | Generic Subprogram | Типовая подпрограмма. Содержит описание переменных, процедур и функций, которые могут быть использованы в нескольких программных приложениях. При этом типовая подпрограмма не содержит определений классов |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_26.png | Generic Package | Типовой пакет. Содержит определение класса, его атрибутов и операций, которое может быть использовано в нескольких программных приложениях |
| http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/50/objects/12/files/12_27.png | Database | База данных. Содержит определение одного или нескольких классов, их атрибутов и, возможно, операций. При этом соответствующие классы могут быть реализованы в форме одной или нескольких таблиц базы данных |

Использование рассмотренных стереотипов существенно увеличивают наглядность графического представления диаграммы компонентов и позволяют архитектору уточнить характер реализации модели программистом на выбранном языке программирования.

**1.Добавление компонента на диаграмму компонентов и редактирование его свойств**

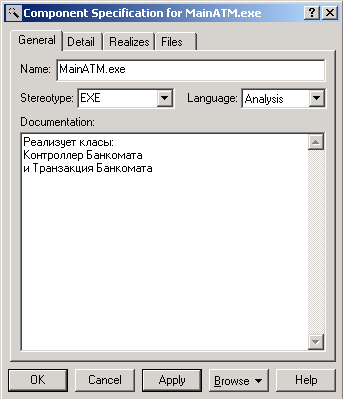
Для добавления компонента на диаграмму компонентов нужно с помощью левой кнопки мыши нажать кнопку с изображением пиктограммы компонента на специальной панели инструментов, отпустить левую кнопку мыши и щелкнуть левой кнопкой мыши на свободном месте рабочего листа диаграммы. Добавить компонент на диаграмму можно также с помощью операции главного меню: **Tools \toCreate \toComponent** или с помощью операции контекстного меню: **New \toComponent**, предварительно выделив представление компонентов в браузере проекта.

В результате этих действий на диаграмме появится изображение компонента с маркерами изменения его геометрических размеров и предложенным средой именем по умолчанию, которое разработчику следует изменить. Продолжая разработку модели системы управления банкоматом, построим для нее диаграмму компонентов. С этой целью изменим имя диаграммы, предложенное по умолчанию Main, на Диаграмма компонентов АТМ, а для первого добавленного компонента зададим имя MainATM.exe (рис. 9.1).

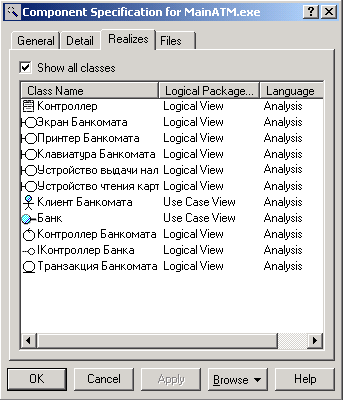


**Рис. 9.1.** Диаграмма компонентов после добавления компонента MainATM.exe

Для каждого компонента можно определить различные свойства, такие как стереотип, язык программирования, декларации, реализуемые классы. Редактирование этих свойств для произвольного компонента осуществляется с помощью диалогового окна спецификации свойств (рис. 9.2).

  
**Рис. 9.2.** Диалоговое окно спецификации свойств компонента MainATM.exe

В частности, для компонента MainATM.exe можно выбрать стереотип <<EXE>> из предлагаемого вложенного списка, поскольку применительно к разрабатываемой модели предполагается реализация этого компонента в форме исполнимого файла. При этом на вкладке Realizes (Реализует) содержатся все классы, включая и актеров, которые на данный момент присутствуют в модели (рис. 9.3). Следует заметить, что классы будут показаны в этом окне только при выбранном свойстве **Show all classes** (Показать все классы).



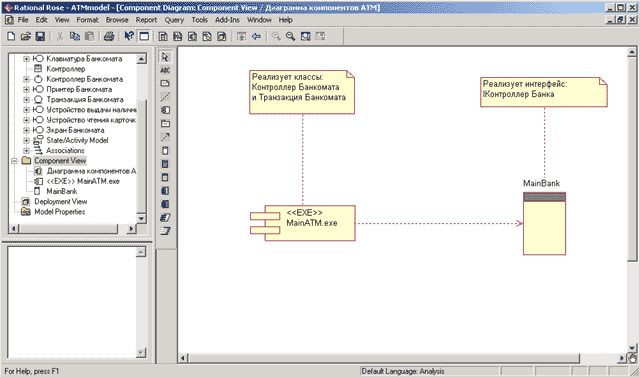
**Рис. 9.3.** Диалоговое окно спецификации свойств компонента MainATM.exe, открытое на вкладке Realizes (Реализует)

По умолчанию в среде IBM Rational Rose для всех добавляемых на диаграмму компонентов в качестве языка реализации используется язык анализа, который в последствии следует изменить на тот язык программирования, который предполагается использовать для написания программного кода. В дальнейшем при генерации программного кода необходимо будет дополнительно выбрать те классы, которые реализует тот или иной компонент модели. Программа IBM Rational Rose поддерживает возможность использования различных языков программирования для реализации различных компонентов модели.

**2.Добавление отношения зависимости и редактирование его свойств**

Добавление отношения зависимости на диаграмму компонентов аналогично добавлению соответствующего отношения на диаграмму вариантов использования. Продолжая разработку модели банкомата, на диаграмму компонентов предварительно следует добавить второй компонент с именем MainBank, для которого выбрать стереотип **Main Program**. Для добавления зависимости между двумя компонентами нужно с помощью левой кнопки мыши нажать кнопку с изображением зависимости на специальной панели инструментов, отпустить левую кнопку мыши, щелкнуть левой кнопкой мыши на изображении исходного компонента на диаграмме и отпустить ее на изображении целевого компонента. В результате этих действий на диаграмме появится изображение отношения зависимости в форме пунктирной линии со стрелкой, соединяющей два выбранных компонента.

Применительно к диаграмме компонентов модели банкомата рассмотренным способом следует добавить отношение зависимости от компонента с именем MainATM.exe к компоненту с именем MainBank. В дополнение к этому для наглядности можно указать в форме примечаний те классы модели, которые предполагается реализовать в данных компонента х (рис. 9.4).



**Рис. 9.4.** Диаграмма компонентов после добавления отношения зависимости между компонентами MainATM.exe и MainBank

Следует заметить, что отношение зависимости в среде IBM Rational Rose не имеет собственного окна спецификации свойств. Именно по этой причине специфицировать свойства данного отношения, такие как имя и стереотип, можно только с помощью текстовой области, что нельзя признать удобным с практической точки зрения.

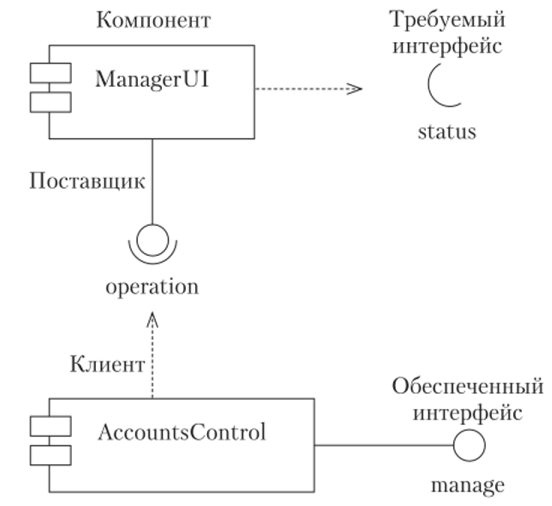


Рис. 9.5 – Диаграмма компонентов

**Задание.**

Изучить теоретические сведения по диаграмме компонентов. Создать данную диаграмму по своей предметной области, ссылаясь на примеры диаграммы (рис. 9.5).

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.

**Лабораторная работа №10. Построение диаграммы Потоков данных**

**Цель работы**: научиться создавать диаграмму кооперации и развертывания.

**Теоретические сведения**

**Диаграммы потоков данных** представляют собой информационную модель (DFD), основными компонентами которой являются различные потоки данных, которые переносят информацию от одной подсистемы к другой. Каждая из подсистем выполняет определенные преобразования входного потока данных и передает результаты обработки информации в виде потоков данных для других подсистем. Основными компонентами диаграмм потоков данных являются: внешние сущности, накопители данных или хранилища, процессы, потоки данных, системы/подсистемы. Внешняя сущность представляет собой материальный объект или физическое лицо, которое может выступать в качестве источника или приемника информации. Примерами внешних сущностей могут служить: клиенты организации, заказчики, персонал, поставщики. Внешняя сущность обозначается прямоугольником с тенью (рисунок 10.1), внутри которого указывается ее имя. При этом в качестве имени рекомендуется использовать существительное в именительном падеже.

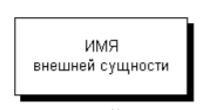


Рисунок 10.1 - Изображение внешней сущности на диаграмме потоков Данных

Процесс представляет собой совокупность операций по преобразованию входных потоков данных в выходные в соответствии с определенным алгоритмом или правилом. Хотя физически процесс может быть реализован различными способами, наиболее часто подразумевается программная реализация процесса. Процесс на диаграмме потоков данных изображается прямоугольником с закругленными вершинами (рисунок 10.2), разделенным на три секции или поля горизонтальными линиями. Поле номера процесса служит для идентификации последнего. В среднем поле указывается имя процесса. В качестве имени рекомендовано использовать глагол в неопределенной форме с необходимыми дополнениями. Нижнее поле содержит указание на способ физической реализации процесса.

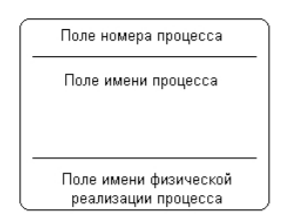


Рисунок 10.2 - Изображение процесса на диаграмме потоков данных

Накопитель данных или хранилище представляет собой абстрактное устройство или способ хранения информации, перемещаемой между процессами. Предполагается, что данные можно в любой момент поместить в накопитель и через некоторое время извлечь, причем физические способы помещения и извлечения данных могут быть произвольными. Накопитель данных на диаграмме потоков данных изображается прямоугольником с двумя полями (рисунок 10.3).



Рисунок 10.3 - Изображение накопителя на диаграмме потоков данных

Поток данных определяет качественный характер информации, передаваемой через некоторое соединение от источника к приемнику. Поток данных на диаграмме DFD изображается линией со стрелкой на одном из ее концов, при этом стрелка показывает направление потока данных. Каждый поток данных имеет свое собственное имя, отражающее его содержание. Таким образом, информационная модель системы в нотации DFD строится в виде диаграмм потоков данных, которые графически представляются с использованием соответствующей системы обозначений.

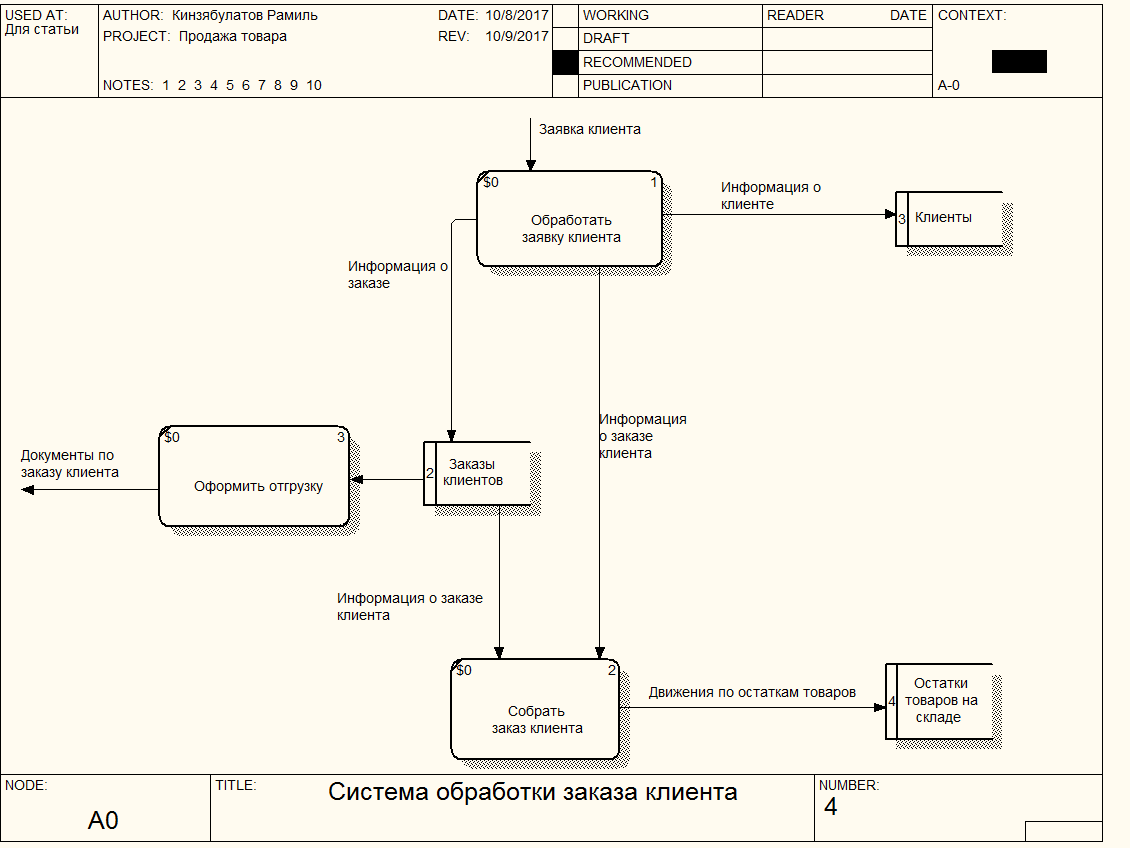


Рисунок 10.4 – Диаграмма потоков данных

**Задание.**

Изучить теоретические сведения по диаграмме потоков данных. Создать данную диаграмму по своей предметной области, ссылаясь на примеры диаграммы (рис. 10.5).

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.

**Лабораторная работа №11. Разработка тестового сценария**

**Цель работы**: усвоить знания о видах тестирования, освоить способы обнаружения и фиксирования ошибок.

**Теоретические сведения**

Общепринятая практика состоит в том, что после завершения продукта и до передачи его заказчику независимой группой тестировщиков проводится тестирование ПО.

Уровни тестирования:

Модульное тестирование. Тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция.

Интеграционное тестирование. Проверяется, есть ли какие – либо проблемы в интерфейсах и взаимодействии между интегрируемыми компонентами, например, не передается информация, передается не корректная информация.

Системное тестирование. Тестируется интегрированная система на ее соответствие исходным требованиям.

Таблица 11.1. – Виды некоторых ошибок и способы их обнаружения.

|  |  |
| --- | --- |
| Виды программных ошибок | Способы их обнаружения |
| Ошибки выполнения, выявляемые автоматически:   * 1. Переполнение, защита памяти;   2. Несоответствие типов;   3. Зацикливание. | Динамический контроль:  Аппаратурой процессора;  Run-time системы программирования;  Операционной системой – по превышению лимита времени |

Тест – это набор контрольных входных данных совместно с ожидаемыми результатами. Тесты должны обладать определенными свойствами.

Детективность: тест должен с большей вероятностью обнаруживать возможные ошибки.

Покрывающая способность: один тест должен выявлять как можно больше ошибок.

Воспроизводимось: ошибка должна выявлять независимо от изменяющихся условий.

С помощью тестирования разных видов обнаруживаются ошибки в разрабатываемом ПО. Для обнаружения ошибок проводится их устранение.

**Задание.**

Создать приложение Простой калькулятор, в котором реализовать выполнение простых операций с вводимыми двумя операндами. Выполнить тестирование приложения на различных данных, отличающихся по типу и значению.

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы. Заполните таблицу тестирования.

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Критерий теста | Результат теста/вид ошибки | Решение по исправлению ошибки |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Лабораторная работа №12. Оценка необходимого количества тестов**

**Цель работы**: получить навыки разработки тестовых сценариев.

**Теоретические сведения**

Оценка времени на работу – сложная задача и многое тут зависит от опыта. Однако, используя необходимые принципы и подходы можно оценить (хотя бы приблизительно) время на выполнение даже новой и неизвестной задачи. Существует множество различных методов оценки от применения математических моделей до комбинирования мнений экспертов.

Предлагаю следующий алгоритм для оценки трудозатрат на тестирование ПО:

1. Декомпозиция требований и определение количества тестов, чтобы их покрыть

2. Определить среднее время на разработку и выполнение 1 теста

3. Определить количество возможных ошибок и время на работу с 1 дефектом

4. Учесть возможные дополнительные затраты времени и риски

5. Получить суммарную оценку, с учетом предыдущих шагов

Далее рассмотрим каждый шаг подробно. При этом в качестве примера возьмем функциональность, для которой по алгоритму будем определять время на тестирование. В качестве функционала рассмотрим простую web форму со строкой ввода, которая позволяет складывать 2 числа:

Требования к системе следующие:

а) Должны суммироваться 2 числа от 0 до 100

б) Для сложения между числами необходимо ввести знак «+»

в) Результат сложения должен отображаться под строкой ввода после нажатия «Calculate»

г) При некорректном вводе и попытке вычисления, под строкой ввода должно появляться сообщение «Ошибка вычисления»

Для упрощения примера мы не будем рассматривать и проверять как и где запускается эта форма, не будем тестировать браузеры, настройки системы и сети, взаимодействие с базой и т.п. В данном примере мы проверяем только ввод\вывод значений и выполнение операции сложения.

Проведем оценку времени на тестирования для нашего примера:

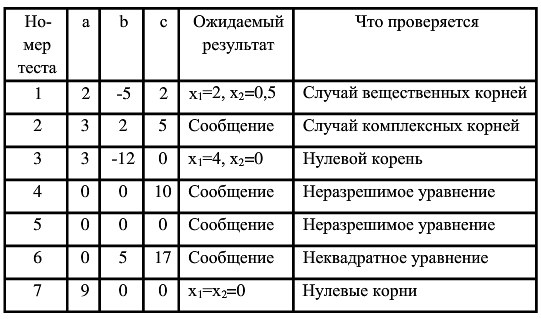
1. Декомпозиция требований и определение количества тестов, чтобы их покрыть

Итак, начнем с анализа и декомпозиции требований. Как это обычно бывает, не все требования явно прописаны в ТЗ, определенная часть условий будет являться следствием перечисленных. Что в итоге мы фактически должны проверить? Ниже я сразу буду указывать краткое описание, какие требования мы проверяем, какие для этого использует тесты и количество необходимых тестов:

|  |  |  |
| --- | --- | --- |
| Что проверяем? | Тесты | Количество тестов |
| Ввод корректных значений в строку ввода и проверка итоговой суммы – ожидаем правильный результат | Вводим в строку 0+100, ожидаем 100 | 1 |
| Вводим в строку 10+0, ожидаем 10 | 1 |
| Вводим в строку 4+5, ожидаем 9 | 1 |
| Вводим новые значения после предыдущего вычисления 66+66, ожидаем 132 | 1 |
| При вводе чисел используем «backspace» и продолжаем ввод, ожидаем правильный результат | 2 |
| Вычисление минимальной и максимальной итоговой суммы – ожидаем правильный результат | Вводим в строку 0+0, ожидаем 0 | 1 |
| Вводим в строку 100+100, ожидаем 200 | 1 |
| Варианты ввода разделителя между числами - ожидаем правильный результат | Вводим в строку 1+2, ожидаем 3 | 1 |
| Вводим в строку 10+20, ожидаем 30 | 1 |
| Вводим в строку 15+25, ожидаем 40 | 1 |
| Варианты ввода разделителя между числами - ожидаем ошибку | Вводим числа без +, используем порядок более 1 пробела | 3 |
| Варианты ввода некорректных символов – ожидаем ошибку | При вводе чисел используем символы «» , . / а и т.д. | 5 |
| Ввод некорректных значений в строку ввода и проверки итоговой суммы – ожидаем ошибку | Вводим в строку -1+100, ожидаем ошибку | 1 |
| Вводим в строку 100, ожидаем ошибку | 1 |
| Вводим в строку 1+101, ожидаем ошибку | 1 |
| Работу кнопки «Calculate» | Нажимаем «Calculate» при пустой строке ввода, ожидаем ошибку | 1 |
| Нажимаем «Calculate» 2 раза после ввода корректных значений, результат не меняется | 1 |
|  | Всего тестов: | 24 |

**Задание.**

* 1. Написать программу решения квадратного уравнения ах2 + bх + с = 0.
  2. Найти минимальный набор тестов для программы нахождения вещественных корней квадратного уравнения ах2 + bх + с = 0. Решение представлено в таблице.



Таким образом, для этой программы предлагается минимальный набор функциональных тестов, исходя из 7 классов выходных данных.

Заповеди по отладки программного средства, предложенные Г. Майерсом.

Заповедь 1. Считайте тестирование ключевой задачей разработки ПС, поручайте его самым квалифицированным и одаренным программистам, нежелательно тестировать свою собственную программу.

Заповедь 2. Хорош тот тест, для которого высока вероятность обнаружить ошибку, а не тот, который демонстрирует правильную работу программы.

Заповедь 3. Готовьте тесты как для правильных, так и для неправильных данных.

Заповедь 4. Документируйте пропуск тестов через компьютер, детально изучайте результаты каждого теста, избегайте тестов, пропуск которых нельзя повторить.

Заповедь 5. Каждый модуль подключайте к программе только один раз, никогда не изменяйте программу, чтобы облегчить ее тестирование.

Заповедь 6. Пропускайте заново все тесты, связанные с проверкой работы какой-либо программы ПС или ее взаимодействия с другими программами, если в нее были внесены изменения (например, в результате устранения ошибки).

* 1. Разработайте набор тестовых сценариев (как позитивных, так и негативных) для следующей программы: Имеется консольное приложение (разработайте самостоятельно). Ему на вход подается строки. На выходе приложение выдает число вхождений второй строки в первую. Например:



Набор тестовых сценариев запишите в виде таблицы, приведенной выше

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.

**Лабораторная работа №13. Разработка тестовых пакетов.**

**Цель работы**: получить навыки разработки тестовых пакетов.

**Теоретические сведения**

В ходе выполнения лабораторной работы провести тестирование по принципу «белого ящика».

Критерии покрытия кода:

* покрытие операторов — каждая ли строка исходного кода была выполнена и протестирована;
* покрытие условий — каждая ли точка решения (вычисления истинно ли или ложно выражение) была выполнена и протестирована;
* покрытие путей — все ли возможные пути через заданную часть кода были выполнены и протестированы;
* покрытие функций — каждая ли функция программы была выполнена;
* покрытие вход/выход — все ли вызовы функций и возвраты из них были выполнены;
* покрытие значений параметров — все ли типовые и граничные значения параметров были проверены.

***Метод покрытия операторов***

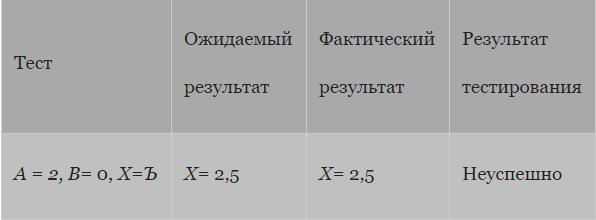
Целью этого метода тестирования является выполнение каждого оператора программы хотя бы один раз.

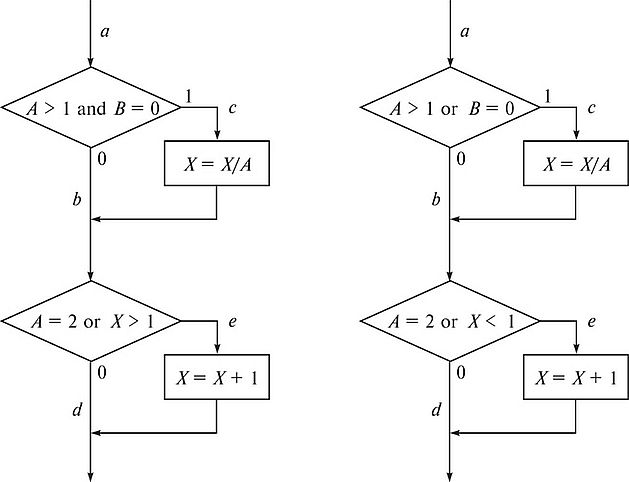
Пример.

Если для тестирования задать значения переменных *А* = 2, *В=* О, *Х=3,* будет реализован путь *асе,* т. е. каждый оператор программы выполнится один раз (рис. 13.1, *а).* Но если внести в алгоритм ошибки — заменить в первом условии *and* на *or,* а во втором *Х>*1 на *Х<*1 (рис. 13.1, *б),* ни одна ошибка не будет обнаружена (табл. Л5.1). Кроме того, путь *abd* вообще не будет охвачен тестом, и если в нем есть ошибка, она также не будет обнаружена. В табл. 13.1 ожидаемый результат определяется по блок-схеме на рис. 13.1, *а,* а фактический — по рис. 13.1, *б.*

Как видно из этой таблицы, ни одна из внесенных в алгоритм ошибок не будет обнаружена.

*Таблица 13.1.* Результат тестирования методом покрытия операторов





*а) б)*

Рис. 13.1. Пример алгоритма программы: *а* — правильный; *б* — с ошибкой

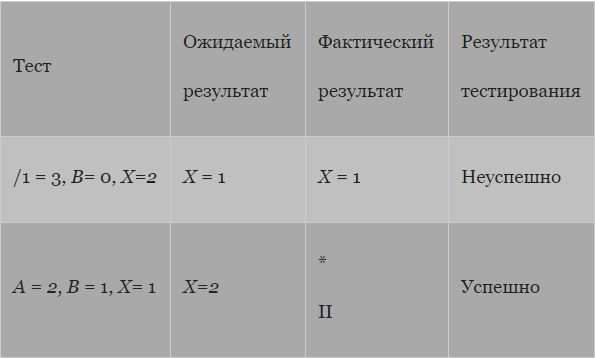
***Метод покрытия решений (покрытия переходов)***

Согласно методу покрытия решений каждое направление перехода должно быть реализовано, по крайней мере, один раз. Этот метод включает в себя критерий покрытия операторов, так как при выполнении всех направлений переходов выполнятся все операторы, находящиеся на этих направлениях.

Для программы, приведенной на рис. 13.1, покрытие решений может быть выполнено двумя тестами, покрывающими пути *{асе, аЬс*!}, либо *{асс1, аЬе*}. Для этого выберем следующие исходные данные; *{А =* 3, *В=* О, *Х=3} —* в первом случае и *{А = 2, В=* 1, *Х=* 1} — во втором. Однако путь, где *X* не меняется, будет проверен с вероятностью 50 %: если во втором условии вместо условия *Х>* записано *Х<,* то ошибка не будет обнаружена двумя тестами.

Результаты тестирования приведены в табл. 13.2.

*Таблица 13.2.* Результат тестирования методом покрытия решений



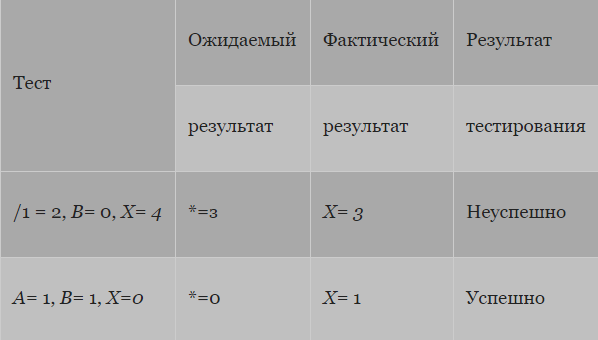
***Метод покрытия условий***

Этот метод может дать лучшие результаты по сравнению с предыдущими. В соответствии с методом покрытия условий записывается число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись, по крайней мере, один раз.

В рассматриваемом примере имеем четыре условия: *{А>* 1, 5=0}, *{А* = 2, *Х>* 1}. Следовательно, требуется достаточное число тестов, такое, чтобы реализовать ситуации, где *А* > 1, *А <* 1, 5 = 0 и й#0 в точке а и /1 = 2, *А \*2, Х>* и Т< 1 в точке *Ь.* Тесты, удовлетворяющие критерию покрытия условий (табл. 13.3), и соответствующие им пути:

* а) *А = 2,* 5 = 0, *Х=4 асе;*
* б) *А =* 1, 5= 1, *Х=0 аbс!.*

*Таблица 13.3.* Результаты тестирования методом покрытия условий



***Метод покрытия решений/условий***

Критерий покрытия решений/условий требует такого достаточного набора тестов, чтобы все возможные результаты каждого условия выполнялись по крайней мере один раз, все результаты каждого решения выполнялись по крайней мере один раз и, кроме того, каждой точке входа передавалось управление по крайней мере один раз.

Недостатки метода:

* не всегда можно проверить все условия;
* невозможно проверить условия, которые скрыты другими условиями;
* недостаточная чувствительность к ошибкам в логических выражениях.

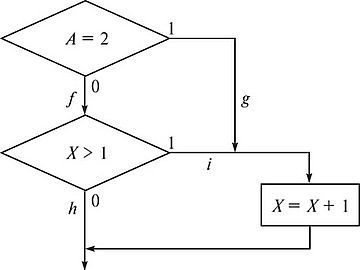
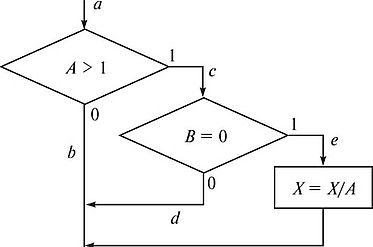
Так, в рассматриваемом примере два теста метода покрытия условий

а) *А* = 2, *В =* О, *Х= 4 асе;*

б*) А=* 1, *В=* 1, *Х=0 аbс!*

отвечают и критерию покрытия решений/условий. Это является следствием того, что одни условия приведенных решении скрывают другие условия в этих решениях. Так, если условие *А>* 1 будет ложным, транслятор может не проверять условия *В=* О, поскольку при любом результате условия *В=* 0 результат решения *((А > )&(В=0))* примет значение *ложь.* То есть в варианте на рис. 13.1 не все результаты всех условий выполнятся в процессе тестирования.

Рассмотрим реализацию того же примера на рис. 13.2. Наиболее полное покрытие тестами в этом случае осуществляется



так, чтобы выполнялись все возможные результаты каждого простого решения. Для этого нужно покрыть пути *aceg* (тест *А = 2, В=* О, *Х=* 4), *асё/к* (тест *А* = 3, *В* = 1, *Х=* 0), *аЬ/к* (тест *А* = 0, *В=* 0, *Х=* 0), *аЬр* (тест Л = 0, *В* = 0, *Х= 2).*

Протестировав алгоритм на рис. 13.2, нетрудно убедиться в том, что критерии покрытия условий и критерии покрытия решений/условий недостаточно чувствительны к ошибкам в логических выражениях.

***Метод комбинаторного покрытия условий***

Критерий комбинаторного покрытия условий удовлетворяет также и критериям покрытия решений, покрытия условий и покрытия решений/условий.

Этот метод требует создания такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении выполнялись по крайней мере один раз. По этому критерию в рассматриваемом примере должны быть покрыты тестами следующие восемь комбинаций:

1. *А>* 1, *В=* 0.*5, А= 2, Х>*1.

2. *А*> 1, *Вф* 0.

3. *А<* 1, *В=* 0.

4. *А<* 1, Вф 0.

6. *А =* 2, 1.

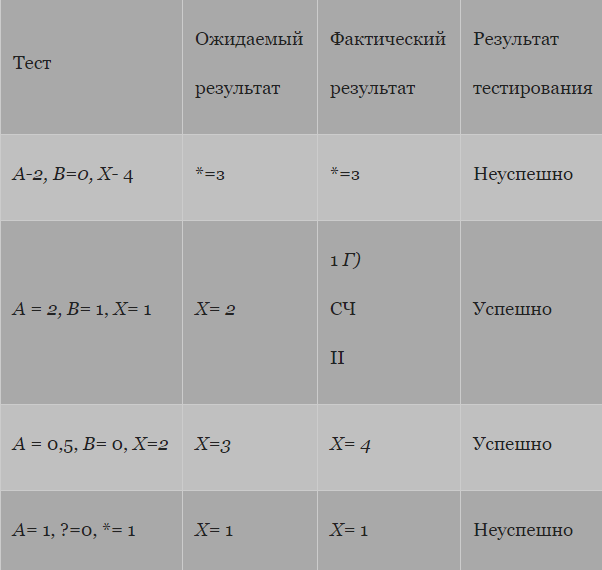
7. Аф2, *Х>* 1.

8. Аф2, *Х<* 1.

Для того чтобы протестировать эти комбинации, необязательно использовать все 8 тестов. Фактически они могут быть покрыты четырьмя тестами (табл. 13.4):

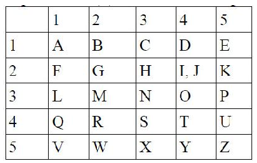
* *А =* 2, *В = 0, Х=4* (покрывает 1,5};
* *А = 2, В = 1, Х= 1* (покрывает 2,6};
* *А =* 0,5, *В=* 0, *Х= 2* (покрывает 3, 7};
* *А = I, В = 0, Х=* (покрывает 4, 8}.

*Таблица 13.4.* Результаты тестирования методом комбинаторного покрытия условий



**Задание.**

**1.** В Древней Греции (II в. до н.э.) был известен шифр, называемый "квадрат Полибия". Шифровальная таблица представляла собой квадрат с пятью столбцами и пятью строками, которые нумеровались цифрами от 1 до 5. В каждую клетку такого квадрата записывалась одна буква. В результате каждой букве соответствовала пара чисел, и шифрование сводилось к замене буквы парой чисел. Для латинского алфавита квадрат Полибия имеет вид:

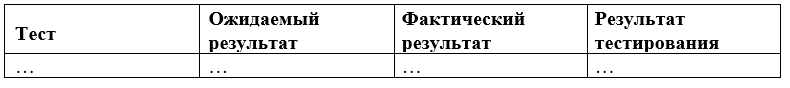


Пользуясь вышеизложенным способом написать программу, которая:

а) зашифрует введенный текст и выведет на экран;

б) считает зашифрованный текст и расшифрует данный текст *(пункт б выполнить по желанию и возможностям).*

***2. Выбрать один из методов.*** Спроектировать тесты по принципу «белого ящика» для программы,разработанной в задании № 1. Выбрать алгоритм для тестирования, обозначить буквами или цифрами ветви этих алгоритмов. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования. Записать тесты, которые позволят пройти по путям алгоритма. Протестировать разработанную вами программу. Результаты оформить в виде таблицы:



**3.**Проверить все виды тестов и сделать выводы об их эффективности.

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.

**Лабораторная работа №14. Оценка программных средств с помощью метрик.**

**Цель работы**: оценка границ времени выполнения проекта, измерить программный продукт и процесс его разработки с использованием метрик.

**Теоретические сведения**

Основной рычаг в планирующих методах — вычисление границ времени выполнения задачи.

Обычно используют следующие оценки:

1. Раннее время начала решения задачи

(при условии, что все предыдущие задачи решены в кратчайшее время).

2. Позднее время начала решения задачи

(еще не вызывает общую задержку проекта).

3. Раннее время конца решения задачи

4. Позднее время конца решения задачи

5. Общий резерв — количество избытков и потерь планирования задач во времени, не приводящих к увеличению длительности критического пути Тк.п.

Все эти значения позволяют руководителю (планировщику) количественно оценить успех в планировании, выполнении задач.

Рекомендуемое правило распределения затрат проекта — 40-20-40:

* на анализ и проектирование приходится 40% затрат (из них на планирование и системный анализ — 5%);
* на кодирование — 20%;
* на тестирование и отладку — 40%.

**Размерно-ориентированные метрики прямо измеряют программный продукт и процесс его разработки.**

Основываются на LOC-оценках (Lines Of Code). LOC-оценка — это количество строк в программном продукте.

Исходные данные для расчета этих метрик сводятся в таблицу (табл.

14.1).

Таблица 14.1. Исходные данные для расчета LOC-метрик

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Проект | Затраты , чел.- мес | Стоимость, тыс. $ | KLOC, тыс.  LOC | Прогр. док- ты, страниц | Ошибки | Люди |
| ааа01 | 24 | 168 | 12.1 | 365 | 29 | 3 |
| bbb02 | 62 | 440 | 27.2 | 1224 | 86 | 5 |
| ссс03 | 43 | 314 | 20.2 | 1050 | 64 | 6 |

На основе таблицы вычисляются размерно-ориентированные метрики производительности и качества (для каждого проекта):

**Функционально-ориентированные метрики** косвенно измеряют программный продукт и процесс его разработки. Вместо подсчета LOC- оценки рассматривается не размер, а функциональность или полезность продукта.

Используется 5 информационных характеристик.

1. Количество внешних вводов. Подсчитываются все вводы пользователя, по которым поступают разные прикладные данные. Вводы должны быть отделены от запросов, которые подсчитываются отдельно.

2. Количество внешних выводов. Подсчитываются все выводы, по которым к пользователю поступают результаты, вычисленные программным приложением. В этом контексте выводы означают отчеты, экраны, распечатки, сообщения об ошибках. Индивидуальные единицы данных внутри отчета отдельно не подсчитываются.

3. Количество внешних запросов. Под запросом понимается диалоговый ввод, который приводит к немедленному программному ответу в форме диалогового вывода. При этом диалоговый ввод в приложении не сохраняется, а диалоговый вывод не требует выполнения вычислений. Подсчитываются все запросы — каждый учитывается отдельно.

4. Количество внутренних логических файлов. Подсчитываются все логические файлы (то есть логические группы данных, которые могут быть частью базы данных или отдельным файлом).

5. Количество внешних интерфейсных файлов. Подсчитываются все логические файлы из других приложений, на которые ссылается данное приложение.

Вводы, выводы и запросы относят к категории транзакция.

*Транзакция* — это элементарный процесс, различаемый пользователем и перемещающий данные между внешней средой и программным приложением.

В своей работе транзакции используют внутренние и внешние файлы. Приняты следующие определения.

*Внешний ввод* — элементарный процесс, перемещающий данные из внешней среды в приложение. Данные могут поступать с экрана ввода или из другого приложения. Данные могут использоваться для обновления внутренних логических файлов. Данные могут содержать как управляющую, так и деловую информацию. Управляющие данные не должны модифицировать внутренний логический файл.

*Внешний вывод* — элементарный процесс, перемещающий данные, вычисленные в приложении, во внешнюю среду. Кроме того, в этом процессе могут обновляться внутренние логические файлы. Данные создают отчеты или выходные файлы, посылаемые другим приложениям. Отчеты и файлы создаются на основе внутренних логических файлов и внешних интерфейсных файлов. Дополнительно этот процесс может использовать вводимые данные, их образуют критерии поиска и параметры, не поддерживаемые внутренними логическими файлами. Вводимые данные поступают извне, но носят временный характер и не сохраняются во внутреннем логическом файле.

*Внешний запрос* — элементарный процесс, работающий как с вводимыми, так и с выводимыми данными. Его результат — данные, возвращаемые из внутренних логических файлов и внешних интерфейсных файлов. Входная часть процесса не модифицирует внутренние логические файлы, а выходная часть не несет данных, вычисляемых приложением (в этом и состоит отличие запроса от вывода).

*Внутренний логический файл* — распознаваемая пользователем группа логически связанных данных, которая размещена внутри приложения и обслуживается через внешние вводы.

*Внешний интерфейсный файл* — распознаваемая пользователем группа логически связанных данных, которая размещена внутри другого приложения и поддерживается им. Внешний файл данного приложения является внутренним логическим файлом в другом приложении.

Каждой из выявленных характеристик ставится в соответствие сложность.

Для этого характеристике назначается низкий, средний или высокий ранг, а затем формируется числовая оценка ранга.

В этой таблице 10 элементов данных: День, Хиты, % от Сумма хитов, Сеансы пользователя, Сумма хитов (по рабочим дням), % от Сумма хитов (по рабочим дням), Сумма сеансов пользователя (по рабочим дням), Сумма хитов (по выходным дням), % от Сумма хитов (по выходным дням), Сумма сеансов пользователя (по выходным дням). Отметим, что поля День, Хиты, % от Сумма хитов, Сеансы пользователя имеют рекурсивные данные, которые в расчете не учитываются.

Таблица 14.2. Пример для расчета элементов данных

|  |  |  |  |
| --- | --- | --- | --- |
| Уровень активности дня недели | | | |
| День | Хиты | % от Сумма хитов | Сеансы пользователя |
| Понедельник | 1887 | 16,41 | 201 |
| Вторник | 1547 | 13,45 | 177 |
| Среда | 1975 | 17,17 | 195 |
| Четверг | 1591 | 13,83 | 191 |
| Пятница | 2209 | 19,21 | 200 |
| Суббота | 1286 | 11,18 | 121 |
| Воскресенье | 1004 | 8,73 | 111 |
| Сумма по рабочим дням | 9209 | 80,08 | 964 |
| Сумма по выходным дням | 2290 | 19,91 | 232 |

Таблица 14.3. Примеры элементов данных

|  |  |
| --- | --- |
| Информационная характеристика | Элементы данных |
| Внешние Вводы | Поля ввода данных, сообщения об ошибках, вычисляемые значения, кнопки |
| Внешние Выводы | Поля данных в отчетах, вычисляемые значения, сообщения об ошибках, заголовки столбцов, которые читаются из внутреннего файла |
| Внешние Запросы | Вводимые элементы: поле, используемое для поиска, щелчок мыши. Выводимые элементы — отображаемые на экране поля |

Таблица 14.4 Правила учета элементов данных из графического интерфейса пользователя

|  |  |
| --- | --- |
| Элемент данных | Правило учета |
| Группа радиокнопок | Т.к. в группе пользователь выбирает только одну радиокнопку, все радиокнопки группы считаются одним элементом данных |
| Группа флажков (переключателей) | Так как в группе пользователь может выбрать несколько флажков, каждый флажок считают элементом данных |
| Командные кнопки | Командная кнопка может определять действие добавления, изменения, запроса. Кнопка ОК может вызывать транзакции (различных типов). Кнопка Next может быть входным элементом запроса или вызывать другую транзакцию. Каждая кнопка считается отдельным элементом данных |
| Списки | Список может быть внешним запросом, но результат запроса может быть элементом данных внешнего ввода |

Таблица 14.5. Ранг и оценка сложности внешних вводов

|  |  |  |  |
| --- | --- | --- | --- |
| Ссылки на файлы | Элементы данных | | |
|  | 1-4 | 5-15 | >15 |
| 0-1 | Низкий (3) | Низкий (3) | Средний (4) |
| 2 | Низкий (3) | Средний (4) | Высокий (6) |
| >2 | Средний (4) | Высокий (6) | Высокий (6) |

Таблица 14.6. Ранг и оценка сложности внешних выводов

|  |  |  |  |
| --- | --- | --- | --- |
| Ссылки на файлы | Элементы данных | | |
|  | 1-4 | 5-19 | >19 |
| 0-1 | Низкий (4) | Низкий (4) | Средний (5) |
| 2-3 | Низкий (4) | Средний (5) | Высокий (7) |
| >3 | Средний (5) | Высокий (7) | Высокий (7) |

Таблица 2.7. Ранг и оценка сложности внешних запросов

|  |  |  |  |
| --- | --- | --- | --- |
| Ссылки на файлы | Элементы данных | | |
|  | 1-4 | 5-19 | >19 |
| 0-1 | Низкий (3) | Низкий (3) | Средний (4) |
| 2-3 | Низкий (3) | Средний (4) | Высокий (6) |
| >3 | Средний (4) | Высокий (6) | Высокий (6) |

Таблица 2.8. Ранг и оценка сложности внутренних логических файлов

|  |  |  |  |
| --- | --- | --- | --- |
| Типы элементов - записей | Элементы данных | | |
|  | 1-19 | 20-20 | >50 |
| 1 | Низкий (7) | Низкий (7) | Средний (10) |
| 2-5 | Низкий (7) | Средний (10) | Высокий (15) |
| >5 | Средний (10) | Высокий (15) | Высокий (15) |

Таблица 2.9. Ранг и оценка сложности внешних интерфейсных файлов

|  |  |  |  |
| --- | --- | --- | --- |
| Типы элементов - записей | Элементы данных | | |
|  | 1-19 | 20-20 | >50 |
| 1 | Низкий (5) | Низкий (5) | Средний (10) |
| 2-5 | Низкий (5) | Средний (7) | Высокий (10) |
| >5 | Средний (7) | Высокий (10) | Высокий (10) |

если во внешнем запросе ссылка на файл используется как на этапе ввода, так и на этапе вывода, она учитывается только один раз. Такое же правило распространяется и на элемент данных (однократный учет).

После сбора всей необходимой информации приступают к расчету метрики — количества функциональных указателей FP (Function Points). Автором этой метрики является А. Албрехт (1979).

Исходные данные для расчета сводятся в табл. 14.10.

Таблица 2.10. Исходные данные для расчета FP-метрик

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя характеристики | Ранг, сложность, количество | | | |
|  | Низкий | Средний | Высокий | Итого |
| Внешние вводы | 0x3 = \_\_ | 0x4 = \_\_ | 0x6 = \_\_ | = 0 |
| Внешние выводы | 0x4 = \_\_ | 0x5 = \_\_ | 0x7 = \_\_ | = 0 |
| Внешние запросы | 0х3 = \_\_ | 0x4 = \_\_ | 0x6 = \_\_ | = 0 |
| Внутренние логические файлы | 0x7 = \_\_ | 0x 10= \_\_ | 0x15 = \_\_ | = 0 |
| Внешние интерфейсные файлы | 0x5 = \_\_ | 0x7 = \_\_ | 0x10 = \_\_ | = 0 |
| Общее количество | | | | = 0 |

В таблицу заносится количественное значение характеристики каждого вида (по всем уровням сложности). Места подстановки значений отмечены прямоугольниками (прямоугольник играет роль метки-заполнителя). Количественные значения характеристик умножаются на числовые оценки сложности. Полученные в каждой строке значения суммируются, давая полное значение для данной характеристики. Эти полные значения затем суммируются по вертикали, формируя общее количество.

Количество функциональных указателей вычисляется по формуле

где Fi — коэффициенты регулировки сложности.

Каждый коэффициент может принимать следующие значения: 0 — нет влияния, 1 — случайное, 2 — небольшое, 3 — среднее, 4 — важное, 5 — основное.

Значения выбираются эмпирически в результате ответа на 14 вопросов, которые характеризуют системные параметры приложения (табл. 14.11).

Таблица 14.11. Определение системных параметров приложения

|  |  |  |
| --- | --- | --- |
| № | Системный параметр | Описание |
| 1 | Передачи данных | Сколько средств связи требуется для передачи или обмена информацией с приложением или системой? |
| 2 | Распределенная обработка данных | Как обрабатываются распределенные данные и функции обработки? |
| 3 | Производительность | Нуждается ли пользователь в фиксации времени ответа или производительности? |
| 4 | Распространенность используемой конфигурации | Насколько распространена текущая аппаратная платформа, на которой будет выполняться приложение? |
| 5 | Скорость транзакций | Как часто выполняются транзакции? (каждый день, каждую неделю, каждый месяц) |
| 6 | Оперативный ввод данных | Какой процент информации надо вводить в режиме онлайн? |
| 7 | Эффективность работы конечного пользователя | Приложение проектировалось для обеспечения эффективной работы конечного пользователя? |
| 8 | Оперативное обновление | Как много внутренних файлов обновляется в онлайновой транзакции? |
| 9 | Сложность обработки | Выполняет ли приложение интенсивную логическую или математическую обработку? |
| 10 | Повторная используемость | Приложение разрабатывалось для удовлетворения требований одного или многих пользователей |
| 11 | Легкость инсталляции | Насколько трудны преобразование и инсталляция приложения? |
| 12 | Легкость эксплуатации | Насколько эффективны и/или автоматизированы процедуры запуска, резервирования и восстановления? |
| 13 | Разнообразные условия размещения | Была ли спроектирована, разработана и поддержана возможность инсталляции приложения в разных местах для различных организаций? |
| 14 | Простота изменений | Была ли спроектирована, разработана и поддержана в приложении простота изменений? |

После вычисления FP на его основе формируются метрики производительности, качества и т. д.:

Область применения метода функциональных указателей — коммерческие информационные системы.

Для продуктов с высокой алгоритмической сложностью используются метрики указателей свойств (Features Points). Они применимы к системному и инженерному ПО, ПО реального времени и встроенному ПО.

Для вычисления указателя свойств добавляется одна характеристика - количество алгоритмов. Алгоритм здесь определяется как ограниченная подпрограмма вычислений, которая включается в общую компьютерную программу.

Примеры алгоритмов: обработка прерываний, инвертирование матрицы, расшифровка битовой строки. Для формирования указателя свойств составляется табл. 14.12.

Таблица 14.12. Исходные данные для расчета указателя свойств

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Характеристика | Кол-во | Сложность | Итого |
| 1 | Вводы | 0 | х4 | = 0 |
| 2 | Выводы | 0 | х5 | = 0 |
| 3 | Запросы | 0 | х4 | = 0 |
| 4 | Логические файлы | 0 | х7 | = 0 |
| 5 | Интерфейсные файлы | 0 | х7 | = 0 |
| 6 | Количество алгоритмов | 0 | х3 | = 0 |
| Общее количество | | | | = 0 |

После заполнения таблицы по формуле (14.1) вычисляется значение указателя свойств.

Для сложных систем реального времени это значение на 25-30% больше значения, вычисляемого по таблице для количества функциональных указателей.

Как показано в табл. 14.13, результаты пересчета зависят от языка программирования, используемого для реализации ПО.

Таблица 14.13. Пересчет FP-оценок в LOC-оценки

|  |  |
| --- | --- |
| Язык программирования | Количество операторов на один FP |
| Ассемблер | 320 |
| С | 128 |
| Кобол | 106 |
| Фортран | 106 |
| Паскаль | 90 |
| C++ | 64 |
| Java | 53 |
| Ada 95 | 49 |
| Visual Basic | 32 |
| Visual C++ | 34 |
| Delphi Pascal | 29 |
| Smalltalk | 22 |
| Perl | 21 |
| HTML3 | 15 |
| LISP | 64 |
| Prolog | 64 |
| Miranda | 40 |
| Haskell | 38 |

**Выполнение оценки в ходе руководства проектом**

Процесс руководства программным проектом начинается с множества действий, объединяемых общим названием планирование проекта. Первое из этих действий — выполнение оценки. Оно закладывает фундамент для других действий по планированию проекта. При оценке проекта чрезвычайно высока цена ошибок. Очень важно провести оценку с минимальным риском.

**Выполнение оценки проекта на основе LOC- и FP-метрик**

Цель этой деятельности — сформировать предварительные оценки, которые позволят:

* предъявить заказчику корректные требования по стоимости и затратам на разработку программного продукта;
* составить план программного проекта.

При выполнении оценки возможны два варианта использования LOC- и FP-данных:

* в качестве оценочных переменных, определяющих размер каждого элемента продукта;
* в качестве метрик, собранных за прошлые проекты и входящих в метрический базис фирмы.

Обсудим шаги процесса оценки.

Шаг 1. Область назначения проектируемого продукта разбивается на

ряд функций, каждую из которых можно оценить индивидуально:

f1, f2,…,fn.

Шаг 2. Для каждой функции fi

, планировщик формирует лучшую

LOCлучшi (FРлучшi), худшую LOCхудшi (FРхудшi) и вероятную оценку

LOCвероятнi (FРвероятнi). Используются опытные данные (из метрического

базиса) или интуиция. Диапазон значения оценок соответствует степени

предусмотренной неопределенности.

 Шаг 3. Для каждой функции/ в соответствии с

 -распределением

вычисляется ожидаемое значение LOC- (или FP-) оценки:

LOCожi =(LOCлучшi + LOCхудшi +4x LOCвероятнi )/ 6.

 Шаг 4. Определяется значение LOC- или FP-производительности

разработки функции.

Используется один из трех подходов:

1) для всех функций принимается одна и та же метрика средней

производительности ПРОИЗВср, взятая из метрического базиса;

2) для i-й функции на основе метрики средней производительности

вычисляется настраиваемая величина производительности:

ПРОИЗВi =ПРОИЗВсрх(LOCср /LOCожi),

где LOCcp — средняя LOC-оценка, взятая из метрического базиса

(соответствует средней производительности);

3) для i-й функции настраиваемая величина производительности

вычисляется по аналогу, взятому из метрического базиса:

ПРОИЗВi =ПРОИЗВанiх(LOCанi

/LOCожi).

Первый подход обеспечивает минимальную точность (при максимальной

простоте вычислений), а третий подход — максимальную точность (при

максимальной сложности вычислений).

 Шаг 5. Вычисляется общая оценка затрат на проект: для первого

подхода

  





n

i

i

1

ЗАТРАТЫ ( LOCо ж )/ПРОИЗВср чел.-мес

;

для второго и третьего подходов

  





n

i

i i

1

ЗАТРАТЫ (LOCо ж /ПРОИЗВ ) чел.-мес .

 Шаг 6. Вычисляется общая оценка стоимости проекта: для первого и

второго подходов

ср

1

СТОИМОСТЬ (LOCо ж ) УД\_СТОИМОСТ Ь



 

n

i

i

,

где УД\_СТОИМОСТЬср — метрика средней стоимости одной строки,

взятая из метрического базиса.

для третьего подхода

СТОИМОСТЬ (LOC УД\_СТОИМОСТ Ь ), ан

1

о ж i

n

i

 i



 

где УД\_СТОИМОСТЬанi — метрика стоимости одной строки аналога,

взятая из метрического базиса. Пример применения данного процесса оценки

приведем ниже.

Пример выполнения оценки проекта на основе LOC- и FP-метрик

Предположим, что поступил заказ от концерна «СУПЕРАВТО». Необходимо

создать ПО для рабочей станции дизайнера автомобиля (РДА). Заказчик

определил проблемную область проекта в своей спецификации:

 ПО РДА должно формировать 2- и 3-мерные изображения для

дизайнера;

 дизайнер должен вести диалог с РДА и управлять им с помощью

стандартизованного графического пользовательского интерфейса;

 геометрические данные и прикладные данные должны содержаться в

базе данных РДА;

 модули проектного анализа рабочей станции должны формировать

данные для широкого класса дисплеев SVGA;

 ПО РДА должно управлять и вести диалог со следующими

периферийными устройствами: мышь, дигитайзер (графический

планшет для ручного ввода), плоттер (графопостроитель), сканер,

струйный и лазерный принтеры.

Прежде всего

-детализировать проблемную область.

-выделить базовые функции ПО

-очертить количественные границы.

-определить, что такое «стандартизованный графический пользовательский

интерфейс», какими должны быть размер и другие характеристики базы

данных РДА и т. д.

Будем считать, что эта работа проделана и что идентифицированы

следующие основные функции ПО:

1. Средства управления пользовательским интерфейсом СУПИ.

2. Анализ двухмерной графики А2Г.

3. Анализ трехмерной графики А3Г.

4. Управление базой данных УБД.

5. Средства компьютерной дисплейной графики КДГ.

6. Управление периферией УП.

7. Модули проектного анализа МПА.

Теперь нужно оценить каждую из функций количественно, с помощью

LOC-оценки.

По каждой функции эксперты предоставляют лучшее, худшее и вероятное

значения. Ожидаемую LOC-оценку реализации функции определяем по

формуле

LOCожi =(LOCлучшi +LOCхудшi +4 х LOCвероятнi)/6,

результаты расчетов заносим в табл. 2.22.

Таблица 2.22. Начальная таблица оценки проекта

Функц

ия

Лучш.

[LOC]

Вероят

.

[LOC]

Худш.

[LOC]

Ожид.

[LOC]

Уд.

стоимость

[$/LОС]

Стоимо

сть[$]

Произв.

[LOC/

[челмес]

Затрат

ы [челмес]

СУПИ 1800 2400 2650 2340

А2Г 4100 5200 7400 5380

АЗГ 4600 6900 8600 6800

УБД 2950 3400 3600 3350

КДГ 4050 4900 6200 4950

УП 2000 2100 2450 2140

МПА 6600 8500 9800 8400

Итого 33360

Для определения удельной стоимости и производительности обратимся в

архив фирмы, где хранятся данные метрического базиса, собранные по уже

выполненным проектам. Предположим, что из метрического базиса

извлечены данные по функциям-аналогам, представленные в табл. 2.23.

Видно, что наибольшую удельную стоимость имеет строка функции

управления периферией (требуются специфические и конкретные знания по

разнообразным периферийным устройствам), наименьшую удельную

стоимость — строка функции управления пользовательским интерфейсом

(применяются широко известные решения).

Таблица 2.23. Данные из метрического базиса фирмы

Функц

ия

LOCанi УД\_СТОИМОСТЬанi[$

/ LOC]

ПРОИЗВанi[L

OC/чел-мес]

СУПИ 585 14 1260

А\_Г 3000 20 440

УБД 1117 18 720

КДГ 2475 22 400

УП 214 28 1400

МПА 1400 18 1800

Считается, что удельная стоимость строки является константой и не

изменяется от реализации к реализации. Следовательно, стоимость

разработки каждой функции рассчитываем по формуле

СТОИМОСТЬi = LOCожi х УД\_СТОИМОСТЬанi

.

Для вычисления производительности разработки каждой функции выберем

самый точный подход — подход настраиваемой производительности:

ПРОИЗВi =ПРОИЗВанi х (LOC анi

/ LOCожi).

Соответственно, затраты на разработку каждой функции будем определять

по выражению

ЗАТРАТЫ i = (LOCожi

/ПРОИЗВi)[чел.-мес].

Теперь мы имеем все необходимые данные для завершения расчетов.

Заполним до конца таблицу оценки нашего проекта (табл. 2.24).

Таблица 2.24. Конечная таблица оценки проекта

Функция Лучш. Вероят. Худш. Ожид.

[LOC]

Уд.

стоимость

[S/LOC]

Стоимость

[$]

Произв.

[LOC/ чел.-

мес]

Затраты

[чел-мес]

СУПИ 1800 2400 2650 2340 14 32760 315 7,4

А2Г 4100 5200 7400 5380 20 107600 245 21,9

A3Г 4600 6900 8600 6800 20 136000 194 35,0

УБД 2950 3400 3600 3350 18 60300 240 13,9

КДГ 4050 4900 6200 4950 22 108900 200 24,7

УП 2000 2100 2450 2140 28 59920 140 15,2

МПА 6600 8500 9800 8400 18 151200 300 28,0

Итого 33360 656680 146

Учитывая важность полученных результатов, проверим расчеты с

помощью FP-указателей. На данном этапе оценивания разумно допустить,

что все информационные характеристики имеют средний уровень сложности.

В этом случае результаты экспертной оценки принимают вид,

представленный в табл. 2.25, 2.26.

Таблица 2.25. Оценка информационных характеристик проекта

Характеристи

ка

Луч

ш.

Веро

ят

Худ

ш.

Ож

ид.

Сложн

ость

Колич

ество

Вводы 20 24 30 24 х 4 96

Выводы 12 15 22 16 х 5 80

Запросы 16 22 28 22 х 4 88

Логические

файлы

4 4 5 4 х 10 40

Интерфейсные

файлы

2 2 3 2 х 7 14

Общее

количество

318

Таблица 2.26. Оценка системных параметров проекта

Коэффициент регулировки

сложности

Оценка

F1 Передачи данных 2

F2 Распределенная обработка данных 0

F3 Производительность 4

F4 Распространенность используемой

конфигурации

3

F5 Скорость транзакций 4

F6 Оперативный ввод данных 5

F7 Эффективность работы конечного

пользователя

5

F8 Оперативное обновление 3

F9 Сложность обработки 5

F10 Повторная используемость 4

F11 Легкость инсталляции 3

F12 Легкость эксплуатации 4

F13 Разнообразные условия размещения 5

F14 Простота изменений 5

Таким образом, получаем:

FР = Общее количество х (0,65+ 0,01 х



14

i 1

Fi

) = 318 x 1,17 = 372.

Используя значение производительности, взятое в метрическом базисе

фирмы,

Производительность = 2,55 [FP / чел.-мес],

вычисляем значения затрат и стоимости:

Затраты = FP / Производительность = 145,9 [чел.-мес],

Стоимость = Затраты х $4500 = $656500.

Итак, результаты проверки показали хорошую достоверность

результатов.

Но мы не будем останавливаться на достигнутом и организуем еще

одну проверку, с помощью модели СОСОМО II.

Примем, что все масштабные факторы и факторы затрат имеют

номинальные значения. В силу этого показатель степени В = 1,16, а

множитель поправки Мp= 1. Кроме того, будем считать, что автоматическая

генерация кода и повторное использование компонентов не

предусматриваются. Следовательно, мы вправе применить формулу

ЗАТРАТЫ = A х РАЗМЕРB

[чел.-мес]

и получаем:

ЗАТРАТЫ = 2,5(33,3)1,16 =145,87 [чел.-мес].

Соответственно, номинальная длительность проекта равна

Длительность = [3,0 х (ЗАТРАТЫ)(0,33+0,2(B-1,01))]=3(145,87)0,36 = 18[мес].

Подведем итоги. Выполнена предварительная оценка программного

проекта. Для минимизации риска оценивания использованы три методики,

доказавшие корректность полученных результатов.

Задание к лабораторной работе.

Для своего программного проекта (для которого написали техническое

задание) сделать оценку границ времени выполнения проекта, измерить

программный продукт и процесс его разработки с использованием размерноориентированных и функционально-ориентированных метрик.

Порядок выполнения работы

1. Вычислить границы времени выполнения задачи: раннее и позднее

времена начала решения задачи, раннее и позднее времена конца

решения задачи, резерв.

2. Измерить программный продукт и процесс его разработки с

использованием размерно-ориентированных метрик.

3. Измерить программный продукт и процесс его разработки с

использованием функционально-ориентированных метрик.

**Задание.**

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.

**Лабораторная работа №15. Инспекция программного кода на предмет соответствия стандартам кодирования.**

**Цель работы**: получить навыки разработки тестовых сценариев.

**Теоретические сведения**

**Задание.**

**Требования к отчету:**

Отчет по лабораторной работе должен содержать краткое описание порядка выполнения работы, так же должны присутствовать скриншоты вашей работы.