МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ИВАНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Шуйский филиал ИвГУ

Кафедра математики, информатики и методики обучения

ОДОБРЕНО:
Руководитель ОП
_____ С.А. Зайцева
(подпись)
«21» июня 2023 г.

Рабочая программа дисциплины
ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Уровень высшего образования:	бакалавриат
Квалификация выпускника:	бакалавр
Направление подготовки:	44.03.05 Педагогическое образование (с двумя профилями подготовки)
Направленность (профиль) образовательной программы:	Математика; Информатика

Шуя

1. Цели освоения дисциплины

Дисциплина «Информационные системы» имеет своей целью содействовать становлению профессиональной компетентности педагога через формирование представления об информационных системах как хранилищах информации, снабженных процедурами ввода, поиска, размещения и выдачи информации, освоение студентами навыков проектирования, разработки и сопровождения современных информационных систем, в том числе в сфере образования.

2. Место дисциплины в структуре ОП

Дисциплина «Информационные системы» является обязательной для изучения; относится к обязательной части блока дисциплин Б1, входит в предметно-методический модуль.

Успешное освоение данной дисциплины будет способствовать готовности обучающихся к успешному прохождению следующих практик: «Производственная практика, научно-исследовательская работа, по предметно-методическому модулю», «Производственная практика, педагогическая (комплексная) практика, по предметно-методическому модулю», подготовке к сдаче и сдаче государственного экзамена, выполнению и защите выпускной квалификационной работы.

Студент, приступающий к изучению дисциплины, должен:

Знать:

- нормативно-правовые акты, регулирующие отношения в сфере проектной и учебно-исследовательской деятельности;
- особенности проектного мышления.

Уметь:

- выделять в поставленной цели основные смысловые и структурные компоненты;
- формулировать задачи на основе этапов получения промежуточных результатов;
- анализировать и практически использовать нормативно-правовые акты в области образования.

Владеть:

- навыками публичного представления результатов решения конкретной задачи проекта;
- способами решения конкретных задач проекта на уровне заявленного качества и за установленное время;
- способами, методами и приемами поиска, анализа и оценки правовой информации, необходимой для постановки и решения профессиональных задач.

3. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

3.1. Компетенции, формированию которых способствует данная дисциплина

При реализации дисциплины формируются следующие компетенции в соответствии с ФГОС ВО по данному направлению подготовки:

а) универсальные компетенции (УК) и индикаторы их достижения:

Наименование категории (группы) универсальных компетенций	Код и наименование универсальной компетенции выпускника	Код и наименование индикатора достижения универсальной компетенции
Разработка и реализация	УК-2. Способен определять круг	УК-2.1. Определяет совокупность

проектов	задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	взаимосвязанных задач, обеспечивающих достижение поставленной цели, исходя из действующих правовых норм. УК-2.2. Определяет ресурсное обеспечение для достижения поставленной цели. УК-2.3. Оценивает вероятные риски и ограничения в решении поставленных задач. УК-2.4. Определяет ожидаемые результаты решения поставленных задач.

б) общепрофессиональные компетенции (ОПК) и индикаторы их достижения:

Наименование категории (группы) общепрофессиональных компетенций	Код и наименование общепрофессиональной компетенции выпускника	Код и наименование индикатора достижения общепрофессиональной компетенции
Правовые и этические основы профессиональной деятельности	ОПК-1. Способен осуществлять профессиональную деятельность в соответствии с нормативными правовыми актами в сфере образования и нормами профессиональной этики	ОПК-1.1. Понимает и объясняет сущность приоритетных направлений развития образовательной системы Российской Федерации, законов и иных нормативно-правовых актов, регламентирующих образовательную деятельность в Российской Федерации, нормативных документов по вопросам обучения и воспитания детей и молодежи, федеральных государственных образовательных стандартов дошкольного, начального общего, основного общего, среднего общего, среднего профессионального образования, профессионального обучения, законодательства о правах ребенка, трудового законодательства. ОПК-1.2. Применяет в своей деятельности основные нормативно-правовые акты в сфере образования и нормы профессиональной этики, обеспечивает конфиденциальность сведений о субъектах образовательных отношений, полученных в процессе профессиональной деятельности.
Контроль и оценка формирования результатов образования	ОПК-5. Способен осуществлять контроль и оценку формирования результатов образования обучающихся, выявлять и корректировать трудности в обучении	ОПК-5.1. Осуществляет выбор содержания, методов, приемов организации контроля и оценки, в том числе ИКТ, в соответствии с установленными требованиями к образовательным результатам обучающихся. ОПК-5.2. Обеспечивает объективность и достоверность оценки образовательных результатов обучающихся. ОПК-5.3. Выявляет и корректирует трудности в обучении, разрабатывает предложения по совершенствованию образовательного процесса.
Взаимодействие с участниками	ОПК-7. Способен взаимодействовать с	ОПК-7.1. Взаимодействует с родителями (законными представителями)

образовательных отношений	участниками образовательных отношений в рамках реализации образовательных программ	обучающихся с учетом требований нормативно-правовых актов в сфере образования и индивидуальной ситуации обучения, воспитания, развития обучающегося. ОПК-7.2. Взаимодействует со специалистами в рамках психолого-медико-педагогического консилиума. ОПК-7.3. Взаимодействует с представителями организаций образования, социальной и духовной сферы, СМИ, бизнес-сообществ и др.

3.2. Планируемые результаты обучения по дисциплине, соотнесенные с индикаторами достижения формируемых компетенций

В результате освоения дисциплины обучающийся должен:

Знать:

– основные этапы проектирования, их последовательность и взаимосвязь (УК-2 (индикаторы – 1, 2, 3, 4);

– современные средства оценивания учебной деятельности и учебных достижений обучающихся (ОПК-5 (индикаторы – 1, 2, 3);

– способы взаимодействия с различными участниками образовательного процесса (ОПК-7 (индикаторы – 1, 2, 3).

Уметь:

– определять совокупность необходимых ресурсов для реализации каждой задачи (УК-2 (индикаторы – 1, 2, 3, 4);

– оценивать уровень и качество каждого ресурса, обеспечивающего выполнение определенной задачи (УК-2 (индикаторы – 1, 2, 3, 4);

– выявлять возможности преодоления рисков и ограничений с учетом имеющихся ресурсов и резервов (УК-2 (индикаторы – 1, 2, 3, 4);

– проектировать процесс решения каждой задачи проекта, выбирая оптимальный способ ее решения, исходя из действующих правовых норм и имеющихся ресурсов и ограничений (УК-2 (индикаторы – 1, 2, 3, 4);

– использовать в образовательном процессе современные средства оценивания учебной деятельности и учебных достижений обучающихся, в том числе электронные (ОПК-5 (индикаторы – 1, 2, 3);

– взаимодействовать с различными участниками образовательных отношений в рамках реализации образовательных программ (ОПК-7 (индикаторы – 1, 2, 3);

– проектировать и обновлять образовательную программу с привлечением обучающихся и их родителей (законных представителей) (ОПК-7 (индикаторы – 1, 2, 3).

Владеть:

– приемами декомпозиции цели, используя вариативные трактовки задач, конкретизирующих различные пути достижения поставленной цели (УК-2 (индикаторы – 1, 2, 3, 4);

– способами определения резервов, использование которых может компенсировать недостаток имеющихся ресурсов (УК-2 (индикаторы – 1, 2, 3, 4);

– способами решения конкретных задач проекта на уровне заявленного качества и за установленное время (УК-2 (индикаторы – 1, 2, 3, 4);

– способами, методами и приемами поиска, анализа и оценки правовой информации, необходимой для постановки и решения профессиональных задач (ОПК-1 (индикаторы – 1,

2);

– способами взаимодействия с различными субъектами образовательного процесса (ОПК-7 (индикаторы – 1, 2, 3).

4. Объём и содержание дисциплины

Очная форма обучения

Объем дисциплины составляет 3 зачётных единицы (81 час.)

Контактная работа 25,83 час.

Самостоятельная работа 55,17 час.

Виды учебных занятий и формы промежуточной аттестации	9 сем.	
	Контактная работа с преподавателем	Самостоятельная работа обучающегося
Лекции	12	30
Лабораторные работы	12	
Групповые консультации перед экзаменом	1,5	-
Экзамен	0,33	25,17
Всего:	25,83	55,17

4.1. Содержание дисциплины по разделам (темам), соотнесенное с видами и объемом учебных занятий

Очная форма обучения

№ п/п	Раздел (тема) дисциплины	Семестр	Виды учебных занятий, включая самостоятельную работу обучающихся и объем (в часах)				Формы текущего контроля успеваемости Формы промежуточной аттестации
			Занятия лекционного типа	Занятия семинарского типа		Самостоятельная работа обучающегося	
				Лаб.	Пр.		
1	Введение в информационные системы. Общая характеристика автоматизированных информационных систем.	9	2			4	доклад
2	Методология и технология разработки информационных систем	9	2			5	доклад
3	Моделирование информационных систем	9	2	4		3	доклад
4	Средства автоматизированного проектирования информационных систем.	9	2	8		6	лабораторная работа
5	Средства разработки программно-информационного ядра информационных систем.	9	2			6	лабораторная работа

6	Современные информационные системы. Особенности построения и эксплуатации геоинформационных (ГИС), систем автоматизированного проектирования (САПР/CAD) и информационных систем управления (ИСУ).	9	2			6	доклад
ИТОГО:			12	12		30	экзамен

4.2. Развернутое описание содержания учебного материала по разделам (темам)

Тема 1. Введение в информационные системы. Общая характеристика автоматизированных информационных систем.

Структура информационной системы. Классификация систем и информационных систем. Классификация информационных систем по признаку структурированности задач. Классификация информационных систем по функциональному признаку.

Понятие информационной технологии управления. Локальные и корпоративные информационные системы. Классификация информационных систем по уровню управления. Системы оперативного уровня. Системы тактического (функционального) уровня. Системы стратегического уровня. Системы поддержки принятия решений. Основные методологии создания экономических информационных управляющих систем.

Тема 2. Методология и технология разработки информационных систем.

Жизненный цикл информационных систем. Принципы и методы создания информационных систем. Модель информационной системы. Информационное обеспечение информационных систем. CASE-средства, их функциональные возможности и характеристика.

Тема 3. Моделирование информационных систем.

Понятие модели предметной области. Структурный подход в моделировании предметной области. Сущность структурного подхода. Методология функционального моделирования SADT. Диаграммы потоков данных DFD. Диаграмма «сущность-связь». Объектно-ориентированный подход в моделировании систем. Сущность объектно-ориентированного подхода. UML — язык универсального моделирования. Диаграммы вариантов использования. Диаграммы деятельности. Диаграммы последовательности. Сравнение структурного и объектно-ориентированного методов моделирования

Тема 4. Средства автоматизированного проектирования информационных систем.

Современные CASE-средства как инструмент многочисленных технологий проектирования информационных систем. Классификация CASE-средств. Характеристики CASE-средств. Функциональный анализ популярных в России CASE-средств.

Этапы анализа предметной области. Анализ деятельности предприятия. Инструменты функционального моделирования бизнес-процессов и использованием стандарта IDEF0. Методология DFD как инструмент моделирования потоков данных. Методология ARIS как инструмент бизнес-моделирования. Язык унифицированного моделирования UML как инструментальное средство моделирования организации и ее бизнес-процессов.

Тема 5. Средства разработки программно-информационного ядра

информационных систем.

Стандартные механизмы доступа к базам данных – Borland Database Engine (BDE), Open Database Connectivity (ODBC), OLEDB, Active XDataObjects (ADO), JavaDataBaseConnectivity (JDBC). Универсальный механизм доступа к данным Universal Data Access как стратегия предоставления доступа к любому типу информации.

СУБД как инструментальное средство разработки информационной системы. Стандарт и реализация языка SQL. Модели данных: реляционная, сетевая, иерархическая. Формы языка SQL. Типы данных SQL. Язык определения данных (DDL). Язык манипулирования данными (DML). Понятие транзакции. Создание объектов базы данных. Ограничения целостности. Клиенты удаленного доступа и построение запросов к СУБД. Технология клиент-сервер. Модели клиент-сервер. Этапы развития серверов баз данных.

Классификация инструментальных средств разработки приложений (средства разработки, ориентированные на конкретные СУБД; средства разработки, универсальные по отношению к СУБД).

Тема 6. Современные информационные системы. Особенности построения и эксплуатации геоинформационных (ГИС), систем автоматизированного проектирования (САПР/CAD) и информационных систем управления (ИСУ).

Современное состояние рынка информационных систем. Понятие, определение и области применения географических информационных систем (ГИС). Классификация ГИС по назначению, архитектуре. Полнофункциональные ГИС (отечественные, зарубежные), характерные свойства полнофункциональных ГИС. Структура подсистем в ГИС. Технические средства ГИС. Информационное обеспечение ГИС. Типы объектов, поддерживаемых в ГИС «MapInfoProfessional». Модели пространственных данных. Понятие слоя, объекта. Системы координат и высот, картографические проекции в ГИС. Источники данных для ГИС. Состав технических средств технологии создания цифровых карт.

Понятие, назначение, классификация и характеристики САПР(CAD). Состав, структура САПР. Зарубежные и отечественные, платные и бесплатные САПР, их сравнительная характеристика. Состав и назначение модулей САПР «nanoCAD» и «КОМПАС».

Планирование потребности в материалах (MaterialRequirementsPlanning, MRP) имощностях (CapacityRequirementsPlanning, CRP). Система MRP II. ERP-системы. BPM (BusinessPerformanceManagement) — автоматизация процессов управленческого планирования и контроля. Аналитическая обработка данных для поддержки принятия решений. OLAP (On-lineAnaliticalProcessing) — оперативный анализ данных. Технологии интеллектуальной обработки данных. Внедрение корпоративных информационных систем. Управление информационными рисками. Экономическая эффективность информационных систем.

Резервное копирование данных. Модели восстановления данных. Технологии и инструменты экспортирования данных и восстановления информации.

Обеспечение безопасности системы. Средства для осуществления атак, системного и сетевого аудита. Защита персональных данных. Средства защиты государственных информационных систем. Средства обеспечения достоверности информации в процессе хранения и обработки.

5. Образовательные технологии, в том числе перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине

В соответствии с требованиями ФГОС ВО в программе данного курса предусмотрено

использовать следующие образовательные технологии: лекции, лабораторные занятия, самостоятельная работа студентов.

В преподавании дисциплины существуют темы, изучение которых должно обязательно начинаться с теоретической подготовки. Чтобы повысить результативность изучения теоретического материала, применяются следующие приемы:

- лекции сопровождаются компьютерными презентациями;
- применяются методики «мозгового штурма», разбора практических ситуаций;
- используются краткие сообщения студентов по новейшим разработкам (по материалам Интернет) с использованием демонстрационных материалов.

Основной и самой результативной формой обучения являются лабораторно-практические занятия. Для преподавания дисциплины создан учебно-методический комплекс, обеспечивающий предоставление информации (электронные учебники и пособия) и управление деятельностью обучаемого.

Разработанные лабораторные задания включают в себя обучающие тексты, набор пошаговых инструкций, учебных задач и заданий, демонстрационный материал.

Такой подход к организации проведения лабораторных занятий стимулирует и организует самостоятельную работу студента. Достигаются не только учебные, но и развивающие цели, т.к. формируется умения по самоорганизации учебной деятельности.

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю): мультимедиа технологии; технологии смешанного обучения.

6. Учебно-методическое обеспечение самостоятельной работы обучающихся

Самостоятельная работа обучающихся по дисциплине осуществляется в следующих формах:

- установка и настройка программного обеспечения и инструментальных средств для разработки информационной системы;
- изучение предметной области проектируемой системы и составление ее модели в словесном виде;
- реализация модели в виде диаграмм и принятых нотаций в применяемой инструментальной среде разработки;
- создание физической модели информационной системы и реализация её инструментальными средствами;
- тестирование и отладка созданной системы, и оценка её эффективности.

Полностью весь методический материал по обеспечению самостоятельной работы студентов приводится в приложении 1 к настоящей программе. Доступ к методическим материалам осуществляется через сайт Шуйского филиала ИвГУ, в том числе через ЭБС Шуйского филиала ИвГУ на платформе «Руконт».

7. Характеристика оценочных средств для текущего контроля успеваемости и промежуточной аттестации по дисциплине

Текущая аттестация осуществляется с использованием накопительного рейтингового оценивания. Критерии оценки деятельности обучающихся разрабатываются в зависимости от уровня их информационно технологической и дидактической подготовки обучающихся.

Для текущего контроля успеваемости применяются:

- лабораторные работы (1-6) на компьютере оцениваются в 5 баллов (30 баллов);
- индивидуальное задание для самостоятельного выполнения (40 баллов);
- доклад по предложенной теме (10 баллов).

Помимо этого учитывается:

- степень активности и самостоятельности обучающегося на лабораторных занятиях – 10 баллов,
- посещаемость (отсутствие пропусков занятий) – 10 баллов.

Промежуточная аттестация по дисциплине (экзамен) проводится с использованием рейтинговой шкалы оценки. На экзамене по дисциплине «Информационные системы» контролируемые знания, умения и владения, характеризующие этапы формирования компетенций в процессе освоения образовательной программы. Экзамен проводится в форме интегрального тестового задания, включающего вопросы по всем темам курса (максимально 30 баллов).

Общая накопительная оценка складывается из баллов текущей успеваемости в семестре по учебной дисциплине и баллов, полученных студентами на экзамене, таким образом, чтобы результирующий балл не превышал значения 100.

Шкала оценки результатов по дисциплине «Информационные системы» при рейтинговой системе оценивания

Повышенный уровень	Базовый уровень	Пороговый уровень	Ниже порогового уровня
5 (отлично)	4 (хорошо)	3 (удовлетворительно)	2 (неудовлетворительно)
85-100 баллов	71-84 балла	61-70 баллов	60 и менее баллов

8. Учебно-методическое и информационное обеспечение дисциплины

Учебная и учебно-методическая литература:

- ***основная литература:***

1. Бройдо, В. Л. Вычислительные системы, сети и телекоммуникации : учебное пособие / В. Л. Бройдо, О. П. Ильина. - 4-е изд. - Санкт-Петербург : Питер, 2011. - 560 с. : ил.
2. Гладких, Т.В. Информационные системы и сети / Т.В. Гладких, Е.В. Воронова ; Министерство образования и науки РФ, Воронежский государственный университет инженерных технологий. – Воронеж : Воронежский государственный университет инженерных технологий, 2016. – 88 с. : схем., ил., табл. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=481994
3. Жданов, С.А. Информационные системы: учебник / С.А. Жданов, М.Л. Соболева, А.С. Алфимова. – Москва : Прометей, 2015. – 302 с.
4. Милехина, О.В. Информационные системы: теоретические предпосылки к построению: учебное пособие / О.В. Милехина, Е.Я. Захарова, В.А. Титова ; Министерство образования и науки Российской Федерации, Новосибирский государственный технический университет. – 2-е изд. – Новосибирск : НГТУ, 2014. – 283 с. : схем., табл. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=258420

- ***дополнительная литература:***

1. Бова, В.В. Основы проектирования математики, информатики и методики обучения / В.В. Бова, Ю.А. Кравченко ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет», Инженерно-технологическая академия. – Ростов-на-Дону ; Таганрог : Издательство Южного федерального университета, 2018. – 106 с. : ил. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=499515
2. Гаскаров, Д. В. Интеллектуальные информационные системы : учебник для вузов / Д. В.

Гаскаров. - Москва : Высшая школа, 2003. - 432 с. : ил.
3. Захарова, Е.Я. Информационные системы: Теоретические предпосылки к построению / Е.Я. Захарова, О.В. Милехина. – Новосибирск : НГТУ, 2010. – 126 с. : табл., схем. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=229309
4. Избачков, Ю. С. Информационные системы : учебник / Ю. С. Избачков, В. Н. Петров. - 2-е изд. - Санкт-Петербург : Питер, 2006. - 656 с. : ил.
5. Информатика. Информационные системы. Информационные технологии. Тестирование. Подготовка к Интернет - экзамену / Г. Н. Хубаев [и др.] ; под общ. ред. Г. Н. Хубаева. - 3-е изд. ; доп. и перераб. - Ростов-на-Дону : Феникс : Март, 2011. - 368 с.
6. Исакова, А.И. Информационные технологии : учебное пособие / А.И. Исакова, М.Н. Исаков ; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Эль Контент, 2012. - 174 с. : ил.,табл., схем. - ISBN 978-5-4332-0036-4 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=208647
7. Лежебоков, А.А. Программные средства и механизмы разработки информационных систем / А.А. Лежебоков ; Министерство образования и науки РФ, Южный федеральный университет, Инженерно-технологическая академия. – Таганрог : Издательство Южного федерального университета, 2016. – 85 с. : схем., ил. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=493216
8. Петров, В. Н. Информационные системы / В. Н. Петров. - Санкт-Петербург : Питер, 2002. - 688 с. : ил.
9. Соболева, М.Л. Информационные технологии : лабораторный практикум / М.Л. Соболева, А.С. Алфимова. - М. : Прометей, 2012. - 48 с.

Интернет-ресурсы:

1. Социальная сеть работников образования. http://nsportal.ru/shkola

Современные профессиональные базы данных и информационно-справочные системы:

1. ЭБС «Университетская библиотека онлайн»;
2. Коллекция Шуйского филиала ИвГУ на национальном цифровом ресурсе «Руконт».
3. Консультант Плюс http://www.consultant.ru/

Программное обеспечение:

В процессе изучения дисциплины используются офисные пакеты Microsoft Office, Open Office Base, свободное программное обеспечение фирмы Oracle, программы моделирование систем StarUMLили VisualUML, RamusEducation, веб-обозреватели, антивирусы, 1С Университет.

9. Материально-техническое обеспечение дисциплины

1. Учебная аудитория для проведения занятий лекционного типа, укомплектованная специализированной мебелью и техническими средствами обучения, служащими для предоставления учебной информации большой аудитории.

2. Лаборатория, оснащенная лабораторным оборудованием, укомплектованная специализированной мебелью и техническими средствами обучения.

3. Учебная аудитория для проведения групповых и индивидуальных консультаций, укомплектованная специализированной мебелью и техническими средствами обучения.

4. Учебная аудитория для проведения текущего контроля и промежуточной

аттестации, укомплектованная специализированной мебелью и техническими средствами обучения.

5. Помещение для самостоятельной работы, оснащенное специализированной мебелью, компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду организации.

6. Учебно-наглядные пособия для занятий лекционного типа, обеспечивающие тематические иллюстрации: презентации и видеоматериалы по тематике: «Информационные системы», «Проектирование информационных систем», «Средства автоматизированного проектирования» и др.

Автор рабочей программы дисциплины: А.В. Ермошин, доцент кафедры математики, информатики и методики обучения, кандидат технических наук, доцент.

Программа рассмотрена на заседании кафедры математики, информатики и методики обучения «15» мая 2019 г., протокол № 8

Программа обновлена
протокол заседания кафедры № 5 от «21» февраля 2020 г.
Согласовано:
Руководитель ОП
_____ С.А. Зайцева
(подпись)

Программа обновлена
протокол заседания кафедры № 10 от «07» июня 2021 г.
Согласовано:
Руководитель ОП
_____ С.А. Зайцева
(подпись)

Программа обновлена
протокол заседания кафедры № 7 от «18» февраля 2022 г.
Согласовано:
Руководитель ОП
_____ С.А. Зайцева
(подпись)

Программа обновлена
протокол заседания кафедры № 11 от «21» июня 2023 г.
Согласовано:
Руководитель ОП
_____ С.А. Зайцева
(подпись)

Приложение 1

Методические указания для обучающихся по освоению дисциплины

На самостоятельную работу студентов по курсу «Информационные системы» отводится более 60% учебного времени от общей трудоемкости курса. Сопровождение самостоятельной работы студентов по данному курсу может быть организовано в следующих формах:

- согласование индивидуальных заданий (виды и темы заданий, сроки представления результатов) самостоятельной работы студента в пределах часов, отведенных на самостоятельную работу;
- консультации (индивидуальные и групповые), в том числе с применением дистанционных технологий взаимодействия;
- промежуточный контроль хода выполнения заданий, строящихся на основе различных способов самостоятельной информационной деятельности в открытой информационной среде.

Преподаватель определяет содержание самостоятельной работы, график её выполнения; создает сетевую информационную и коммуникационную среду для организации самостоятельной работы.

В процессе организации самостоятельной работы особое внимание уделяется не только развитию информационной культуры, но и формированию готовности к кооперации, к работе в коллективе в сетевом пространстве, для чего целесообразно стимулировать студентов к совместной деятельности в малых группах.

Одной из важнейших особенностей заданий для самостоятельной работы по курсу «Информационные системы» является их ориентация на формирование способности оценивать приоритеты и ограничения при выборе средств информационных технологий для решения управленческих задач, а также на мотивирование самообразовательной деятельности с использованием ИКТ.

К типовым заданиям для самостоятельной работы студентов относятся:

− Конспектирование и реферирование литературных источников по тематике учебного курса.
− Решение задач, связанных с установкой программного обеспечения для проектирования и разработки информационных систем;
− Изучение и анализ предложенных электронных образовательных ресурсов, содержащих информацию по темам дисциплины;
− Подготовка к тестированию на занятиях по каждой теме курса;
− Повторение материала с использованием конспектов.

Перечень вопросов для самостоятельного изучения

1. Этапы развития информационных систем.
2. Понятия информационной системы. Основные задачи информационных систем.
3. Основные свойства и процессы в информационных системах. Пользователи информационных систем.
4. Структура информационной системы.
5. Классификация систем и информационных систем.
6. Классификация информационных систем по признаку структурированности задач.
7. Классификация информационных систем по функциональному признаку и уровням управления.

8. Жизненный цикл информационных систем.
9. Принципы и методы создания информационных систем. Модель информационной системы.
10. Информационное обеспечение информационных систем.
11. CASE-средства, их функциональные возможности и характеристика.
12. Области применения и примеры реализации информационных систем. Требования, предъявляемые к информационным системам.
13. Лингвистическое обеспечение информационных систем. Роль и логика языковых средств поиска документальной информации.
14. Семантика информационных систем, основанных на концепции баз данных. Идентификация и поиск информации.
15. Интерфейсные средства информационного поиска.

Задания для организации лабораторных занятий:

ЛАБОРАТОРНАЯ РАБОТА № 1. ОСОБЕННОСТИ РЕАЛИЗАЦИИ ЯЗЫКА UML BCASE-ИНСТРУМЕНТАРИИ VISUALUML

Цель работы: закрепление теоретических сведений об унифицированном языке UML; изучение средства визуального моделирования объектно-ориентированных информационных систем Visual UML; получение практических навыков работы с программным средствомVisua lUML.

Краткие теоретические сведения

CASE-средства (от ComputerAidedSoftware/SystemEngineering) позволяют проектировать любые системы на компьютере. Как необходимый элемент системного и структурно-функционального анализа, CASE-средство позволяет моделировать бизнес-процессы, базы данных, компоненты программного обеспечения, деятельность и структуру организаций. CASE-средства применяются практически во всех сферах деятельности, и их использование оптимизирует системы, снижает расходы, повышает эффективность, снижает вероятности ошибок.

Программное средство Visual UML– это пакет, являющийся фактическим стандартом в области UML-проектирования.

Visual UML– средство визуального моделирования объектно-ориентированных информационных систем. Работа продукта основана на универсальном языке моделирования UML.

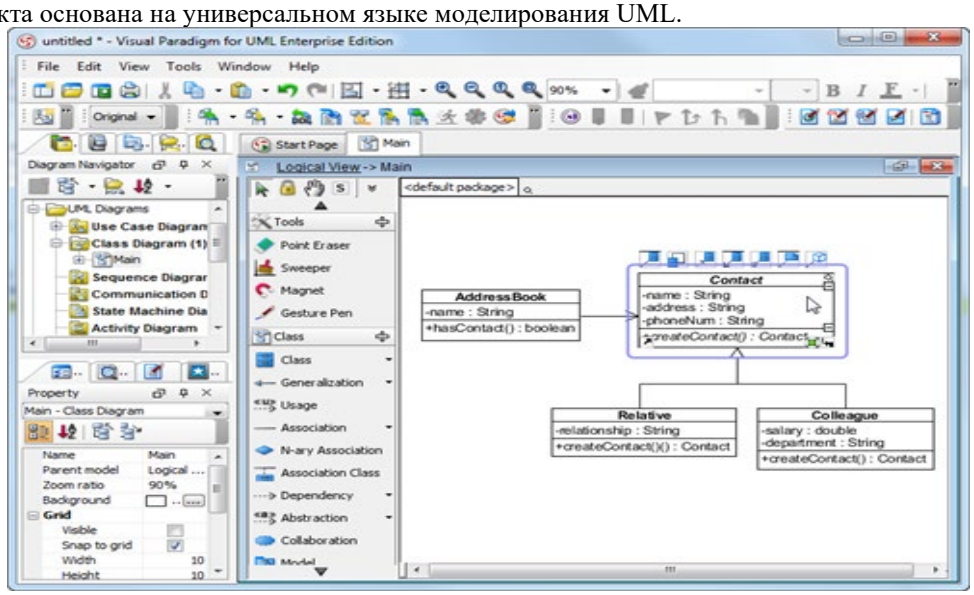


Рис. 1. Программный продукт Visual UML 5.3

Мощный, многофункциональный UML-совместимый набор инструментов объектного моделирования, обеспечивающий полную и всестороннюю поддержку для всех типов диаграмм.

Visual UML включает UML расширения для BusinessObject моделирования, моделирования сетевых приложений (при помощи Conallen WAE), XML (XSD) моделирования, и моделирования данных/баз данных.

Visual UML может использоваться для моделирования многих типов систем, включая базы данных,

информационные системы и веб-приложения. Плюс, всесторонняя поддержка для Activity, Collaborations, State и Sequence диаграмм. Visual UML может использоваться для моделирования систем в режиме реального времени.

Программа Visual UML доступна в трех версиях:

- Standard Edition
- Developer Edition
- Plus-Developer Edition (with VBA)

Программа Visual UML имеет встроенный словарь моделей и данных, а также обладает современным, интуитивным пользовательским интерфейсом. Данный программный продукт предоставляет возможность редактор диаграмм с MDI, а также настройку меню и инструментальных панелей. В Visual UML представлен модельный глоссарий для каждой диаграммы.

Все модельные элементы, диаграммы и объекты диаграмм могут иметь ограничения, тэги, требования, сценарии, и тестирование. Все это реализовано в VisualUML. Данный программный продукт позволяет экспортировать и импортировать модели (включая диаграммы) в различные форматы файлов, такие как EMF и WMF, BMP, JPEG, TIFF, PCX, GIF, HTML и Windows Clipboard.

Порядок выполнения работы

1. Изучить теоретические сведения по теме: “Особенности реализации UMLв CASE-инструментарии Visual UML ”.
2. Открыть CASE-средство Visual UML, изучите его основные компоненты.
3. Ответить на контрольные вопросы.

Контрольные вопросы

1. Дайте определение понятию «язык UML». Назовите основные этапы развития языка UML.
2. Что такое пакет, приведите пример графического изображения пакетов.
3. Что такое CASE-средства. Перечислите, какие вы знаете CASE-средства?
4. Для чего предназначено программное обеспечение Visual UML?

Критерии оценки:
Всего баллов – 5 (Максимально)
Выполнение заданий лабораторной работы может быть оценено от 0 до 3 баллов.
Ответы на контрольные вопросы – от 0 до 2 баллов.

ЛАБОРАТОРНАЯ РАБОТА № 2. ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Цель работы: закрепление теоретических сведений о диаграмме вариантов использования; овладение практическими навыками моделирования процессов, описывающих взаимодействие объектов в диаграмме вариантов использования.

Краткие теоретические сведения

Диаграммы вариантов использования описывают функциональное назначение системы или то, что система должна делать в процессе своего функционирования.

Диаграмма вариантов использования (usecasediagram) — диаграмма, на которой изображаются отношения между актерами и вариантами использования.

Создание диаграммы вариантов использования имеет следующие цели:

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
- сформулировать общие требования к функциональному поведению системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Назначение диаграммы: проектируемая программная система представляется в форме вариантов использования, с которыми взаимодействуют внешние сущности или актеры.

Диаграмма вариантов использования представляет собой граф специального вида, который является графической нотацией для представления конкретных вариантов использования, актеров и отношений между этими элементами.

Базовыми элементами диаграммы вариантов использования являются вариант использования и актер.

Вариант использования (usecase) — внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами.

Вариант использования представляет собой спецификацию общих особенностей поведения или функционирования моделируемой системы без рассмотрения внутренней структуры этой системы.

Каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим актером, сами эти действия не изображаются на рассматриваемой диаграмме.

Содержание варианта использования может быть представлено в форме дополнительного пояснительного текста, который раскрывает смысл или семантику действий при выполнении данного варианта использования. Такой пояснительный текст получил название *текста-сценария* или просто *сценария*.

Вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его имя в форме существительного (рис. 2, а) или глагола (рис. 2, б) с пояснительными словами. Сам текст имени должен начинаться с заглавной буквы.

Имя - строка текста, которая используется для идентификации любого элемента модели.

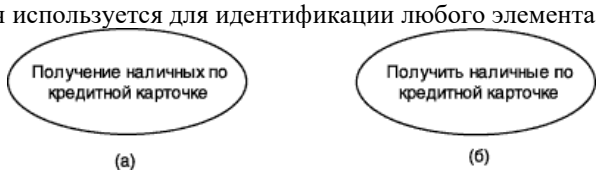


Рис. 2 Графическое обозначение варианта использования

Каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемая система по запросу актера, т. е. определяет один из способов применения системы. Сервис, который инициализируется по запросу актера, должен представлять собой законченную последовательность действий (после того как система закончит обработку запроса актера, она должна возвратиться в исходное состояние, в котором снова готова к выполнению следующих запросов).

Диаграмма вариантов использования содержит конечное множество вариантов использования, которые в целом должны определять все возможные стороны ожидаемого поведения системы. Для удобства множество вариантов использования может рассматриваться как отдельный пакет.

Актер (actor) — согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними.

Актер представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения задач.

Каждый актер может рассматриваться как некая отдельная роль относительно конкретного варианта использования. Стандартным графическим обозначением актера на диаграммах является фигурка "человечка", под которой записывается имя актера (рис. 3).



Рис. 3 Графическое обозначение актера

В некоторых случаях актер может обозначаться в виде прямоугольника класса со стереотипом <<actor>> и обычными составляющими элементами класса. Имена актеров должны начинаться с заглавной буквы и следовать рекомендациям использования имен для типов и классов модели. При этом символ отдельного актера связывает соответствующее описание актера с конкретным именем.

Актеры используются для моделирования внешних по отношению к проектируемой системе сущностей, которые взаимодействуют с системой. В качестве актеров могут выступать другие системы, в том числе подсистемы проектируемой системы или ее отдельные классы. Наиболее наглядный пример актера — конкретный посетитель web-сайта в Интернет со своими параметрами аутентификации.

Актеры взаимодействуют с системой посредством передачи и приема сообщений от вариантов использования. Сообщение представляет собой запрос актером сервиса от системы и получение этого сервиса. Это взаимодействие может быть выражено посредством ассоциаций между отдельными актерами и вариантами использования.

Кроме этого, с актерами могут быть связаны **интерфейсы**, которые определяют, каким образом другие элементы модели взаимодействуют с этими актерами (например, датчик, устройство считывания штрих-кода и т.д.).

Интерфейс обозначается в виде маленького круга, рядом с которым записывается его имя. В качестве имени может быть существительное или строка текста. Если имя записывается на английском языке, то оно

начинается с заглавной буквы I.

Графический символ интерфейса соединяется с тем вариантом использования, который его поддерживает, на диаграмме двумя способами:

- сплошной линией (указывает, что связанный с интерфейсом вариант использования должен реализовывать все необходимые для него сервисы);
- пунктирной линией со стрелкой (означает, что вариант использования предназначен для спецификации только того сервиса, который необходим для реализации данного интерфейса).

Примечание (note) предназначено для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта (комментарии разработчика - дата; ограничения - на значения отдельных связей или экземпляры сущностей; помеченные значения).

Графически обозначаются прямоугольником с "загнутым" верхним правым уголком (рис. 4). Примечание может относиться к любому элементу диаграммы, в этом случае их соединяет пунктирная линия. Если примечание относится к нескольким элементам, то от него проводятся несколько линий.



Рис. 4. Примеры примечаний на диаграммах вариантов использования

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

Между элементами диаграммы вариантов использования могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров и вариантов. Один актер может взаимодействовать с несколькими вариантами использования. В свою очередь один вариант использования может взаимодействовать с несколькими актерами, предоставляя для всех них свой сервис.

В языке UML имеется несколько стандартных видов отношений между актерами и вариантами использования:

- ассоциации (associationrelationship);
- включения (includerelationship);
- расширения (extendrelationship);
- обобщения (generalizationrelationship).

Общие свойства вариантов использования могут быть представлены тремя различными способами: с помощью отношений включения, расширения и обобщения.

Отношение ассоциации используется при построении всех графических моделей систем. Применительно к диаграммам вариантов использования ассоциация служит для обозначения конкретной роли актера при его взаимодействии с отдельным вариантом использования.

На диаграмме вариантов использования, так же как и на других диаграммах, отношение ассоциации обозначается сплошной линией между актером и вариантом использования. Эта линия может иметь некоторые дополнительные обозначения, например, имя и кратность (рис. 5).

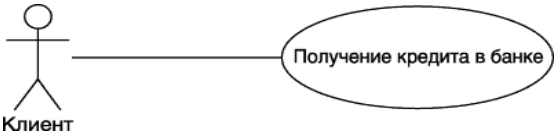


Рис. 5 Пример графического представления отношения ассоциации между актером и вариантом использования

Отношение включения (include) — это разновидность отношения зависимости между базовым вариантом использования и его специальным случаем.

Отношение включения устанавливается между двумя вариантами использования и указывает на то, что заданное поведение для одного варианта использования включается в качестве составного фрагмента в последовательность поведения другого варианта использования.

Графически отношение включения обозначается в форме пунктирной линии со стрелкой, направленной от базового варианта использования к включаемому варианту использования. При этом данная линия помечается стереотипом <<include>>, как показано на рис. 6.

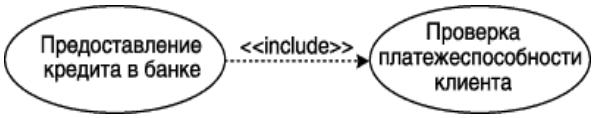


Рис. 6. Пример графического изображения отношения включения между вариантами использования

Отношение включения, направленное от варианта использования "Предоставление кредита в банке" к варианту использования "Проверка платежеспособности клиента", указывает на то, что каждый экземпляр первого варианта использования всегда включает в себя функциональное поведение или выполнение второго варианта использования. В этом смысле поведение второго варианта использования является частью поведения первого варианта использования на данной диаграмме.

Один вариант использования может входить в несколько других вариантов, а также содержать в себе другие варианты. Включаемый вариант использования является независимым от базового варианта в том смысле, что он предоставляет последнему инкапсулированное поведение, детали реализации которого скрыты от последнего и могут быть легко перераспределены между несколькими включаемыми вариантами использования. Базовый вариант зависит только от результатов выполнения включаемого в него варианта использования, но не от структуры включаемых в него вариантов.

Отношение расширения (extend) определяет взаимосвязь базового варианта использования с другим вариантом использования, функциональное поведение которого задействуется базовым не всегда, а только при выполнении дополнительных условий.

Графически обозначается в форме пунктирной линии со стрелкой, направленной от того варианта использования, который является расширением для базового варианта использования. Данная линия со стрелкой должна быть помечена стереотипом <<extend>>, как показано на рис. 7.

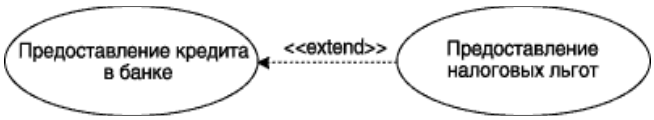


Рис. 7. Пример графического изображения отношения расширения между вариантами использования

В изображенном фрагменте имеет место отношение расширения между базовым вариантом использования "Предоставление кредита в банке" и вариантом использования "Предоставление налоговых льгот". Это означает, что свойства поведения первого варианта использования в некоторых случаях могут быть дополнены функциональностью второго варианта использования. Для того чтобы это расширение имело место, должно быть выполнено определенное логическое условие данного отношения расширения.

Если базовый вариант использования выполняет некоторую последовательность действий, которая определяет его поведение, и при этом имеется точка расширения на экземпляр другого варианта использования, то проверяется логическое условие данного отношения. Если это условие выполняется, исходная последовательность действий расширяется посредством включения действий другого варианта использования. Условие отношения расширения проверяется один раз — при первой ссылке на точку расширения, и если оно выполняется, то все расширяющие варианты использования вставляются в базовый вариант.

Отношение обобщения используется тогда, когда два и более актера могут иметь общие свойства, т. е. взаимодействовать с одним и тем же множеством вариантов использования одинаковым образом. Такая общность свойств и поведения представляется в виде отношения обобщения с другим актером, который моделирует соответствующую общность ролей.

Графически обозначается сплошной линией со стрелкой в форме не закрашенного треугольника, которая указывает на родительский вариант использования (рис. 8). Эта линия со стрелкой имеет специальное название — стрелка-обобщение.

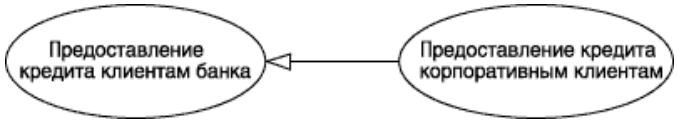


Рис. 8. Пример графического изображения отношения обобщения между вариантами использования

В данном примере отношение обобщения указывает на то, что вариант использования "Предоставление кредита корпоративным клиентам" - специальный случай варианта использования "Предоставление кредита клиентам банка". Другими словами, первый вариант использования является специализацией второго варианта использования.

Вариант использования "Предоставление кредита клиентам банка" еще называют *предком* или родителем по отношению к варианту использования "Предоставление кредита корпоративным клиентам", а последний - *потомком* по отношению к первому варианту использования.

Потомок наследует все свойства поведения своего родителя, а также может обладать дополнительными особенностями поведения.

Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми особенностями поведения родительских вариантов и участвуют во всех отношениях родительских вариантов.

Язык UML включает в себя еще и дополнительные графические обозначения, ориентированные для решения задач из определенной предметной области. Примеры подобных обозначений, которые используются для моделирования бизнес-систем и могут быть изображены на диаграммах вариантов использования: бизнес-актер, сотрудник и бизнес - вариант использования.

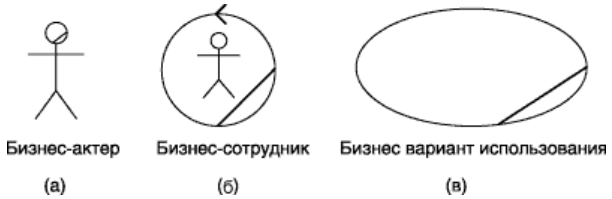


Рис. 9. Графические изображения бизнес-актера (а), бизнес-сотрудника (б) и бизнес-варианта использования (в)

Применение этих элементов графической нотации иллюстрирует пример представления диаграмм вариантов использования для системы продажи товаров в супермаркете.

Одно из главных назначений диаграммы вариантов использования заключается в формализации функциональных требований к системе. Общее количество актеров в модели не должно превышать 20, а вариантов использования -50, в противном случае модель теряет свою наглядность.

Для разработки диаграммы вариантов использования рекомендуется некоторая последовательность действий:

- определить главных и второстепенных актеров;
- определить цели главных актеров по отношению к системе;
- сформулировать основные варианты использования, которые специфицируют функциональные требования к системе;
- упорядочить варианты использования по степени убывания риска их реализации;
- рассмотреть все базовые варианты использования в порядке убывания их степени риска;
- выделить участников, интересы, условия выполнения выбранного варианта использования;
- написать сценарий реализации выбранного варианта использования;
- определить исключения в выполнении сценария варианта использования;
- написать сценарии для всех исключений;
- выделить общие варианты использования и изобразить их взаимосвязи с базовыми со стереотипом <<include>>;
- выделить варианты использования для исключений и изобразить их взаимосвязи с базовыми со стереотипом <<extend>>;
- проверить диаграмму на отсутствие дублирования вариантов использования и актеров.

Пример: Программное средство представляет собой базу данных «Автоматизация процесса составления расписания в учебном заведении». Программное средство обеспечивает корректировку данных, а именно в БД «Группы» может изменяться перечень предметов в соответствие с курсом группы и отделением; в БД «Предметы» могут изменяться номера аудиторий и фамилии преподавателей; осуществляет поиск по ФИО преподавателя, номеру аудитории, названию предмета и номеру группы. Программное средство составляет расписание работы для конкретного преподавателя на неделю; для группы на неделю; для группы по конкретному предмету, а также отчет о загрузке аудиторий на каждый день.

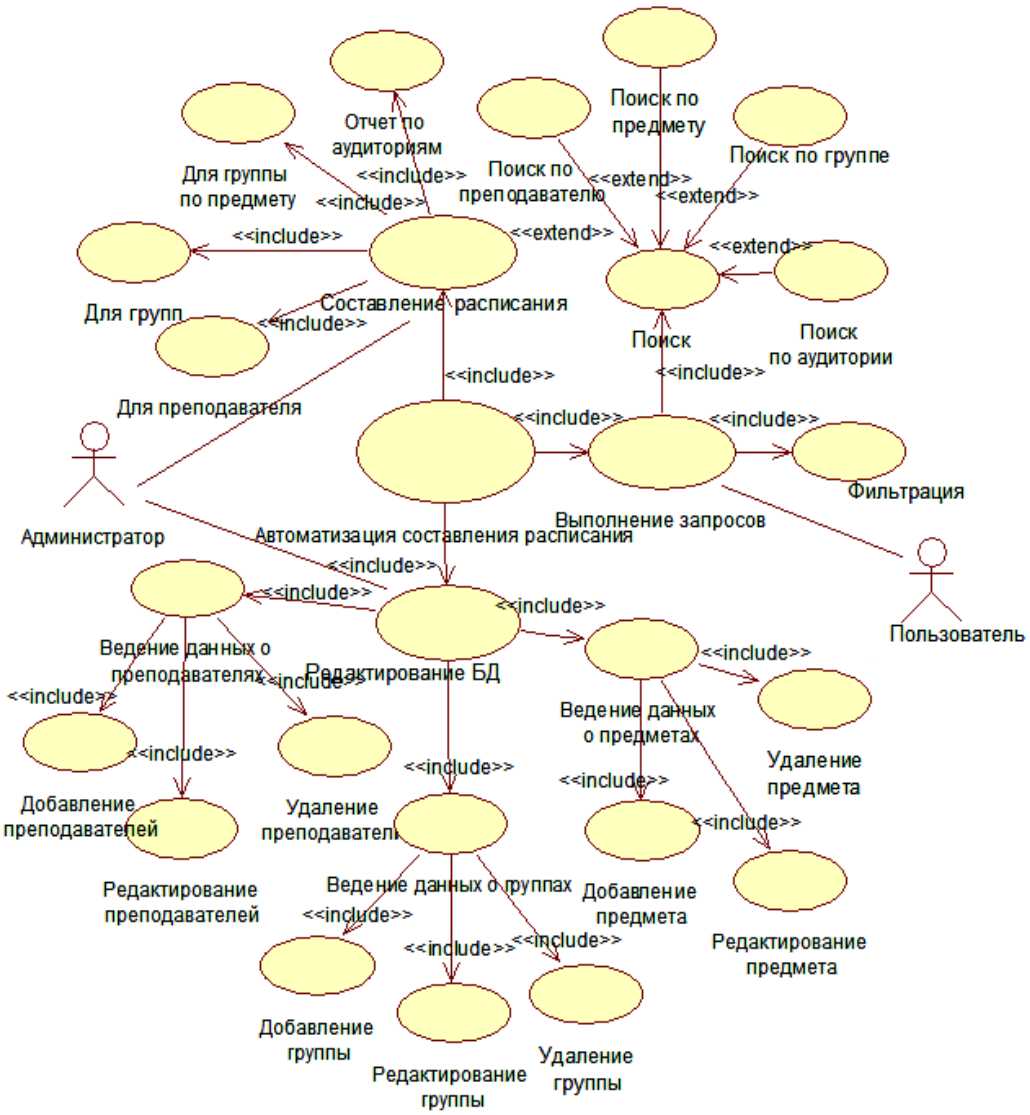


Рис. 10. Пример диаграммы вариантов использования

Порядок выполнения работы

1. Изучить теоретические сведения по теме “Диаграмма вариантов использования”.
2. Разработать диаграмму вариантов использования для произвольной системы своего индивидуального задания (Приложение А).
3. Оформить отчет, включив в него описание всех компонентов диаграммы вариантов использования согласно индивидуальному варианту задания, ответы на контрольные вопросы.

Контрольные вопросы

1. Дайте определение понятию «диаграмма вариантов использования».
2. Опишите назначение диаграммы вариантов использования.
3. Дайте определение следующим понятиям «актер», «вариант использования», «интерфейс», «сценарий».
4. Дайте определение и классификацию отношений в диаграмме вариантов использования.
5. В чем заключается отношение включения?
6. В чем заключается отношение ассоциации?
7. В чем заключается отношение обобщения?
8. В чем заключается отношение расширения?

Критерии оценки:
Всего баллов – 5 (Максимально)

Выполнение заданий лабораторной работы может быть оценено от 0 до 3 баллов.
Ответы на контрольные вопросы – от 0 до 2 баллов.

ЛАБОРАТОРНАЯ РАБОТА № 3. ДИАГРАММА КЛАССОВ

Цель работы: закрепление теоретических сведений о диаграмме классов; овладение практическими навыками моделирования процессов, описывающих взаимодействие объектов в диаграмме классов.

Краткие теоретические сведения

Диаграмма классов отражает различные взаимосвязи между отдельными сущностями предметной области, а также описывает их внутреннюю структуру и типы отношений.

Диаграмма классов (classdiagram) — диаграмма языка UML, на которой представлена совокупность декларативных или статических элементов модели, таких как классы с атрибутами и операциями, а также связывающие их отношения.

Диаграмма классов предназначена для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

Диаграмма классов содержит интерфейсы, пакеты, отношения и даже отдельные экземпляры классификаторов, такие как объекты и связи.

Класс (class) — абстрактное описание множества однородных объектов, имеющих одинаковые атрибуты, операции и отношения с объектами других классов.

Графически изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции (рис. 11). В этих секциях могут указываться имя класса, атрибуты и операции класса. Даже если секции атрибутов и операций пусты, в обозначении класса они должны быть выделены горизонтальной линией.



Рис. 11. Варианты графического изображения класса на диаграмме классов

В первом случае для класса Окружность (рис. 12, а) указаны только его атрибуты – точка на координатной плоскости, которая определяет расположение ее центра. Для класса Окно (рис. 12, б) указаны только его операции, при этом секция его атрибутов оставлена пустой. Для класса Счет (рис. 12, в) дополнительно изображена четвертая секция, в которой указано требование – реализовать резервное копирование объектов этого класса.

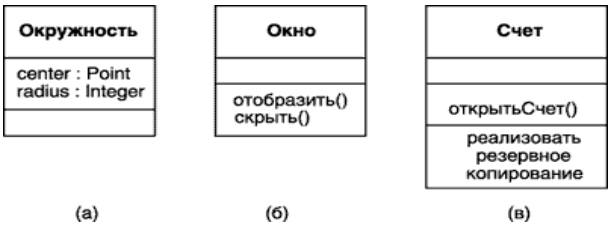


Рис. 12. Примеры графического изображения конкретных классов

Имя класса должно быть уникальным в пределах пакета, который может содержать одну или несколько диаграмм классов. Имя класса записывается по центру секции имени полужирным шрифтом и должно начинаться с заглавной буквы.

В секции имени класса могут также находиться стереотипы или ссылки на стандартные шаблоны, от которых образован данный класс и, соответственно, от которых он наследует атрибуты и операции. Здесь также могут записываться и другие общие свойства этого класса.

Класс может иметь или не иметь экземпляров или объектов. В зависимости от этого в языке UML

различают конкретные и абстрактные классы.

Конкретный класс (concreteclass) — класс, на основе которого могут быть непосредственно созданы экземпляры или объекты.

Рассмотренные выше обозначения относятся к конкретным классам.

Абстрактный класс (abstractclass) — класс, который не имеет экземпляров или объектов.

Для обозначения имени абстрактного класса используется курсив.

Когда необходимо явно указать, к какому пакету относится тот или иной класс, используется специальный символ разделитель – двойное двоеточие - (::). Синтаксис строки имени класса в этом случае будет следующий:

`<Имя пакета>::<Имя класса>`

Например, если определен пакет с именем Банк, то класс Счет в этом банке может быть записан в виде: Банк::Счет.

Атрибут (attribute) — содержательная характеристика класса, описывающая множество значений, которые могут принимать отдельные объекты этого класса.

Атрибут класса служит для представления отдельного свойства или признака, который является общим для всех объектов данного класса. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, его исходного значения.

Общий формат записи отдельного атрибута класса следующий:

`<квантор видимости><имя атрибута> [кратность] :<тип атрибута> = <исходное значение> {строка-свойство}`

Видимость (visibility) — качественная характеристика описания элементов класса, характеризующая потенциальную возможность других объектов модели оказывать влияние на отдельные аспекты поведения данного класса.

Видимость в языке UML специфицируется с помощью *квантора видимости* (может быть опущен), который может принимать одно из 4-х возможных значений и отображаться при помощи специальных символов.

- "+" – область видимости типа общедоступный (public). Атрибут с этой областью видимости доступен или виден из любого другого класса пакета, в котором определена диаграмма.
- "#" – область видимости типа защищенный (protected). Атрибут недоступен или невиден для всех классов, за исключением подклассов данного класса.
- "-" – область видимости типа закрытый (private). Атрибут недоступен или невиден для всех классов без исключения.
- "~" – область видимости типа пакетный (package). Атрибут недоступен или невиден для всех классов за пределами пакета, в котором определен класс-владелец данного атрибута.

Вместо условных графических обозначений можно записывать соответствующее ключевое слово: public, protected, private, package.

Имя атрибута используется в качестве идентификатора соответствующего атрибута и поэтому должно быть уникальным в пределах данного класса. Имя атрибута - обязательный элемент, должно начинаться со строчной (малой) буквы и не должно содержать пробелов.

Кратность (multiplicity) — спецификация области значений допустимой мощности, которой могут обладать соответствующие множества.

Кратность атрибута характеризует общее количество конкретных атрибутов данного типа, входящих в состав отдельного класса:

`[нижняя граница .. верхняя граница].`

Интервалов кратности для отдельного атрибута может быть несколько. Если в качестве кратности указывается единственное число, то кратность атрибута принимается равной данному числу. Если же указывается "*", то кратность атрибута может быть произвольным положительным целым числом или нулем.

Тип атрибута представляет собой выражение, семантика которого определяется некоторым типом данных, определенным в пакете Типы данных языка UML или самим разработчиком, или в зависимости от языка программирования, который предполагается использовать для реализации данной модели.

Исходное значение служит для задания начального значения соответствующего атрибута в момент создания отдельного экземпляра класса. Если оно не указано, то значение соответствующего атрибута не определено на момент создания нового экземпляра класса.

При задании атрибутов могут быть использованы дополнительные синтаксические конструкции — это подчеркивание строки атрибута, пояснительный текст в фигурных скобках и косая черта перед именем атрибута. Подчеркивание строки атрибута означает, что соответствующий атрибут общий для всех объектов данного класса, т.е. его значение у всех создаваемых объектов одинаковое (аналог ключевого слова static в некоторых языках программирования).

Пояснительный текст в фигурных скобках может означать две различные конструкции. Если в этой строке

имеется знак равенства, то вся запись Строка-свойство служит для указания дополнительных свойств атрибута, которые могут характеризовать особенности изменения значений атрибута в ходе выполнения программы. Фигурные скобки как раз и обозначают фиксированное значение соответствующего атрибута для класса в целом, которое должны принимать все вновь создаваемые экземпляры класса без исключения. Это значение принимается за исходное значение атрибута, которое не может быть переопределено в последующем.

Отсутствие строки-свойства по умолчанию трактуется так, что значение соответствующего атрибута может быть изменено в программе.

Знак "/" перед именем атрибута указывает на то, что данный атрибут является производным от некоторого другого атрибута этого же класса.

Производный атрибут(derivedelement) — атрибут класса, значение которого для отдельных объектов может быть вычислено посредством значений других атрибутов этого же объекта.

Операция (operation) - это сервис, предоставляемый каждым экземпляром или объектом класса по требованию своих клиентов, в качестве которых могут выступать другие объекты, в том числе и экземпляры данного класса.

Общий формат записи отдельной операции класса следующий:

```
<квантор видимости><имя операции>(список параметров):
<выражение типа возвращаемого значения> {строка-свойство}
```

Квантор видимости операции аналогичен квантору видимости атрибутов.

Имя операции – обязательный элемент, должно начинаться со строчной (малой) буквы, и записываться без пробелов.

Список параметров является перечнем разделенных запятой формальных параметров, каждый из которых, в свою очередь, может быть представлен в следующем виде:

```
<направление параметра><имя параметра> :<выражение типа> = <значение параметра по умолчанию>
```

Параметр (parameter) — спецификация переменной операции, которая может быть изменена, передана или возвращена.

Параметр может включать имя, тип, направление и значение по умолчанию. Направление параметра — есть одно из ключевых слов in, out или inout со значением in по умолчанию, в случае если вид параметра не указывается. Имя параметра есть идентификатор соответствующего формального параметра, при записи которого следуют правилам задания имен атрибутов.

Выражение типа возвращаемого значения указывает на тип данных значения, которое возвращается объектом после выполнения соответствующей операции. Две точки и выражение типа возвращаемого значения могут быть опущены, если операция не возвращает никакого значения. Для указания нескольких возвращаемых значений данный элемент спецификации операции может быть записан в виде списка отдельных выражений.

Операция с областью действия на весь класс показывается подчеркиванием имени и строки выражения типа. В этом случае под областью действия операции понимаются все объекты этого класса. В этом случае вся строка записи операции подчеркивается.

Строка-свойство служит для указания значений свойств, которые могут быть применены к данной операции. Строка-свойство может отсутствовать, если свойства не специфицированы.

Список формальных параметров и тип возвращаемого значения не обязателен.

В рамках профиля для процесса разработки ПО (The UML ProfileforSoftwareDevelopmentProcesses) языка UML можно рассмотреть три специальных графических примитива, которые могут быть использованы для уточнения семантики отдельных классов при построении различных диаграмм:

Управляющий класс (controlclass) — класс, отвечающий за координацию действий других классов. На каждой диаграмме классов должен быть хотя бы один управляющий класс, причем количество посылаемых объектам управляющего класса сообщений мало, по сравнению с числом рассылаемых ими.

Управляющий класс отвечает за координацию действий других классов. Как правило, данный класс является активным и инициирует рассылку множества сообщений другим классам модели. Кроме специального обозначения управляющий класс может быть изображен в форме прямоугольника класса со стереотипом <<control>> (рис. 13, а).

Класс-сущность (entityclass) — пассивный класс, информация о котором должна храниться постоянно и не уничтожаться с выключением системы. Класс-сущность содержит информацию, которая должна храниться постоянно и не уничтожается с уничтожением объектов данного класса или прекращением работы моделируемой системы, связанные с выключением системы или завершением программы.

Как правило, класс-сущность соответствует отдельной таблице базы данных. В этом случае его атрибуты являются полями таблицы, а операции – присоединенными или хранимыми процедурами. Этот класс пассивный и лишь принимает сообщения от других классов модели. Класс-сущность может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<entity>> (рис. 13, б).

Граничный класс (boundaryclass) — класс, который располагается на границе системы с внешней средой и непосредственно взаимодействует с актерами, но является составной частью системы. Граничный класс может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<boundary>> (рис. 13, в).

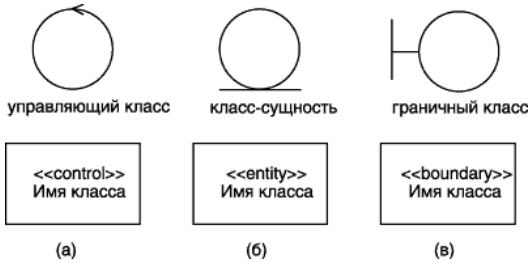


Рис. 13. Графическое изображение классов для моделирования ПО

В рамках профиля для бизнес-моделирования (The UML ProfileforBusinessModeling) языка UML модно рассмотреть три специальных графических примитива, которые могут быть использованы для уточнения семантики отдельных классов при построении моделей бизнес-систем:

Сотрудник (businessworker) — класс, служащий на диаграмме классов для представления любого сотрудника, который является элементом бизнес-системы и взаимодействует с другими сотрудниками при реализации бизнес-процесса. Этот класс также может быть изображен в форме прямоугольника класса со стереотипом <<worker>> или <<internalWorker>> (рис. 14, а).

Сотрудник для связи с окружением (caseworker) – класс, служащий для представления в бизнес-системе такого сотрудника, который, являясь элементом бизнес-системы, непосредственно взаимодействует с актерами (бизнес-актерами) при реализации бизнес-процесса. Этот класс также может быть изображен в форме прямоугольника класса со стереотипом <<caseWorker>> (рис. 14, б).

Бизнес-сущность (businessentity) — специальный случай класса-сущности, который также не инициирует никаких сообщений. Этот класс служит для сохранения информации о результатах выполнения бизнес-процесса в моделируемой бизнес-системе или организации. Этот класс также может быть изображен в форме прямоугольника класса со стереотипом <<businessentity>> (рис. 14, в).

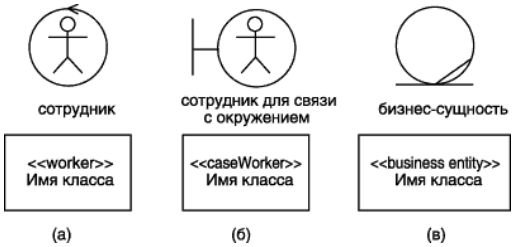


Рис. 14. Графическое изображение классов для моделирования бизнес-систем

Кроме внутреннего устройства классов важную роль при разработке проектируемой системы имеют различные отношения между классами, которые также могут быть изображены на диаграмме классов. Базовые отношения, изображаемые на диаграммах классов:

- отношение ассоциации (associationrelationship);
- отношение обобщения (generalizationrelationship);
- отношение агрегации (aggregationrelationship);
- отношение композиции (compositionrelationship).

Отношение ассоциации соответствует наличию произвольного отношения или взаимосвязи между классами. Оно обозначается сплошной линией со стрелкой или без нее и с дополнительными символами, которые характеризуют специальные свойства ассоциации.

Ассоциации применительно к диаграммам классов немного шире, чем применительно к диаграммам вариантов использования. В качестве дополнительных специальных символов могут использоваться имя ассоциации, символ навигации, а также имена и кратность классов-ролей ассоциации. Имя ассоциации - необязательный элемент ее обозначения. Однако если оно задано, то записывается с заглавной буквы рядом с линией ассоциации. Отдельные классы ассоциации играют определенную роль в соответствующем отношении, на что явно указывает имя концевых точек ассоциации на диаграмме. Наиболее простой случай данного отношения - *бинарная ассоциация*, которая служит для представления произвольного отношения между двумя классами. Она связывает в точности два различных класса и может быть ненаправленным (симметричным) или

направленным отношением. Частный случай бинарной ассоциации – *рефлексивная ассоциация*, которая связывает класс с самим собой. Ненаправленная бинарная ассоциация изображается линией без стрелки. Для нее на диаграмме может быть указан порядок чтения классов с использованием значка в форме треугольника рядом с именем данной ассоциации.

В качестве простого примера ненаправленной бинарной ассоциации можно рассмотреть отношение между двумя классами - классом «Компания» и классом «Сотрудник» (рис. 15). Они связаны между собой бинарной ассоциацией «Работает». Для данного отношения определен следующий порядок чтения следования классов - сотрудник работает в компании.

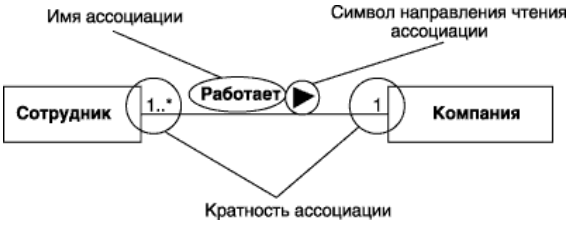


Рис. 15. Графическое изображение ненаправленной бинарной ассоциации между классами

Направленная бинарная ассоциация изображается сплошной линией с простой стрелкой на одной из ее концевых точек. Направление этой стрелки указывает на то, какой класс является первым, а какой - вторым.

В качестве простого примера направленной бинарной ассоциации можно рассмотреть отношение между двумя классами - классом «Клиент» и классом «Счет» (рис. 16). Они связаны между собой бинарной ассоциацией с именем «Имеет», для которой определен порядок следования классов. Это означает, что конкретный объект класса «Клиент» всегда должен указываться первым при рассмотрении взаимосвязи с объектом класса «Счет», например, <клиент, счет_1, счет_2,…, счет_n>.

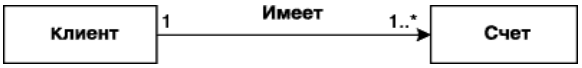


Рис. 16. Графическое изображение направленной бинарной ассоциации между классами

Частный случай отношения ассоциации - так называемая *исключающая ассоциация* (Xor-association). Семантика данной ассоциации указывает на то, что из нескольких потенциально возможных вариантов данной ассоциации в каждый момент времени может использоваться только один.

На диаграмме классов исключающая ассоциация изображается пунктирной линией, соединяющей две и более ассоциации (рис. 6.7), рядом с которой записывается ограничение в форме строки текста в фигурных скобках: {xor}.

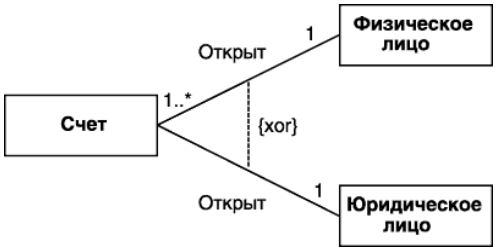


Рис. 17. Графическое изображение исключающей ассоциации между тремя классами

Тернарная ассоциация связывает отношением три класса. Ассоциация более высокой арности называется n -арной ассоциацией.

n-арная ассоциация- ассоциация между тремя и большим числом классов.

Каждый экземпляр такой ассоциации представляет собой упорядоченный набор (кортеж), содержащий n экземпляров из соответствующих классов. Такая ассоциация связывает отношением более чем три класса, при этом класс может участвовать в ассоциации более чем один раз.

Графически n -арная ассоциация обозначается ромбом, от которого ведут линии к символам классов данной ассоциации. Имя n -арной ассоциации записывается рядом с ромбом соответствующей ассоциации. Однако порядок классов в n -арной ассоциации, в отличие от порядка множеств в отношении, на диаграмме не фиксируется.

В качестве примера тернарной ассоциации можно рассмотреть отношение между тремя классами: «Сотрудник», «Компания» и «Проект». Данная ассоциация указывает на наличие отношения между этими тремя классами, которое может представлять информацию о проектах, реализуемых в компании, и о

сотрудниках, участвующих в выполнении отдельных проектов (рис. 18).

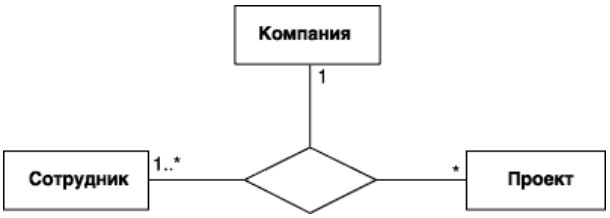


Рис. 18. Графическое изображение тернарной ассоциации между тремя классами

Класс может быть присоединен к линии ассоциации пунктирной линией, т.е. обеспечивать поддержку свойств соответствующей n-арной ассоциации, а сама n-арная ассоциация имеет атрибуты, операции и/или ассоциации. Такой класс является ассоциативным классом.

Класс ассоциация(associationclass) - модельный элемент, который одновременно является ассоциацией и классом.

Отдельный класс в ассоциации может играть определенную роль в данной ассоциации. Поэтому вводиться специальный элемент - *концевая точка ассоциации*, которая графически соответствует точке соединения линии ассоциации с отдельным классом.

Конец ассоциации(associationend) - концевая точка ассоциации, которая связывает ассоциацию с классификатором. Конец ассоциации - часть ассоциации, но не класса. Каждая ассоциация может иметь два или больше концов ассоциации.

Роль (role) - имеющее имя специфическое поведение некоторой сущности, рассматриваемой в определенном контексте. Роль может быть статической или динамической.

Имя роли представляет собой строку текста рядом с концом ассоциации для соответствующего класса. Она указывает на специфическую роль, которую играет класс, являющийся концом рассматриваемой ассоциации. Имя роли не обязательный элемент обозначений.

Кратность ассоциации относится к концам ассоциации и обозначается в виде интервала целых чисел, аналогично кратности атрибутов и операций классов, но без прямых скобок. Этот интервал записывается рядом с концом соответствующей ассоциации и означает потенциальное число отдельных экземпляров класса, которые могут иметь место, когда остальные экземпляры или объекты классов фиксированы.

Так, для примера (рис. 18) кратность " 1 " для класса «Компания» означает, что каждый сотрудник может работать только в одной компании. Кратность " 1..* " для класса «Сотрудник» означает, что в каждой компании могут работать несколько сотрудников, общее число которых заранее неизвестно и ничем не ограничено. Вместо кратности " 1..* " нельзя записать только символ " * ", поскольку последний означает кратность " 0..* ". Для данного примера это означало бы, что отдельные компании могут совсем не иметь сотрудников в своем штате. Такая кратность приемлема в ситуациях, когда в компании вообще не выполняется никаких проектов.

Имя ассоциации рассматривается в качестве отдельного атрибута у соответствующих классов ассоциаций и может быть указано на диаграмме явным способом в определенной секции прямоугольника класса.

Ассоциация является наиболее общей формой отношения в языке UML. Все другие типы рассматриваемых отношений можно считать частным случаем данного отношения.

Отношение обобщения является отношением классификации между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком).

Менее общий элемент модели должен быть согласован с более общим элементом и может содержать дополнительную информацию. Отношение обобщения описывает иерархическое строение классов и наследование их свойств и поведения.

Согласно одному из главных принципов методологии ООАП - наследованию, класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет собственные свойства и поведение, которые могут отсутствовать у класса-предка.

Отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов (рис. 19). Стрелка указывает на более общий класс (класс-предок или суперкласс), а ее начало - на более специальный класс (класс-потомок или подкласс).

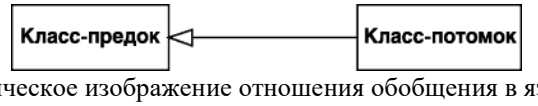
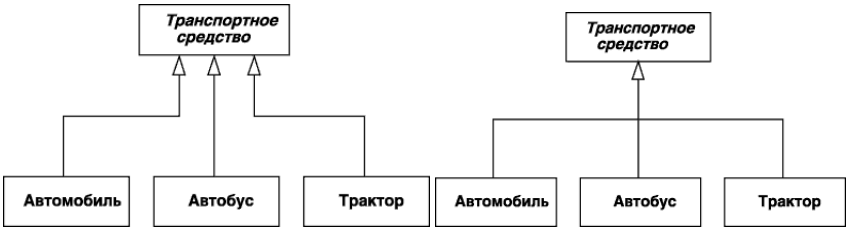


Рис. 19. Графическое изображение отношения обобщения в языке UML

От одного класса-предка одновременно могут наследовать несколько классов-потомков. т.е. в класс-предок входит несколько линий со стрелками.

Пример: класс «Транспортное средство» (курсив обозначает абстрактный класс) может выступать в качестве суперкласса для подклассов, соответствующих конкретным транспортным средствам, таким как: «Автомобиль», «Автобус», «Трактор» и другим (рис 20).



а) б)

Рис. 20. а) Пример графического изображения отношения обобщения для нескольких классов-потомков; б) Альтернативный вариант графического изображения отношения обобщения классов для случая объединения отдельных линий

Графическое изображение отношения обобщения по форме соответствует иерархическому дереву, где класс-предок является корнем дерева, а классы-потомки - его листьями.

В дополнение к простой стрелке обобщения может быть присоединена строка текста, указывающая на специальные свойства этого отношения в форме ограничения. Этот текст будет относиться ко всем линиям обобщения, которые идут к классам-потомкам. Поскольку отмеченное свойство касается всех подклассов данного отношения, именно поэтому спецификация этого свойства осуществляется в форме ограничения, которое должно быть записано в фигурных скобках.

В качестве ограничений могут быть использованы следующие ключевые слова языка UML:

- {complete} - означает, что в данном отношении обобщения специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может.
- {incomplete} - означает случай, противоположный первому. А именно, предполагается, что на диаграмме указаны не все классы-потомки. В последующем возможно разработчик восполнит их перечень, не изменяя уже построенную диаграмму.
- {disjoint} - означает, что классы-потомки не могут содержать объектов, одновременно являющихся экземплярами двух или более классов.
- {overlapping} - случай, противоположный предыдущему. А именно, предполагается, что отдельные экземпляры классов-потомков могут принадлежать одновременно нескольким классам.

С учетом дополнительного использования стандартного ограничения диаграмма классов (рис. 20) может быть уточнена (рис. 21).

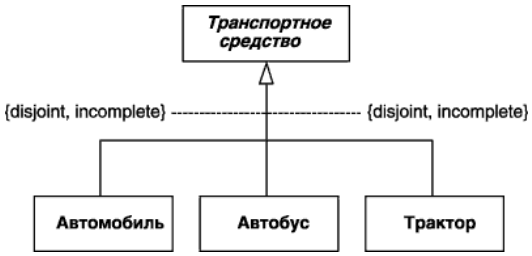


Рис. 21. Вариант уточненного графического изображения отношения обобщения классов с использованием строки-ограничения

Отношение агрегации имеет место между несколькими классами в том случае, если один из классов представляет собой сущность, которая включает в себя в качестве составных частей другие сущности. Данное отношение применяется для представления системных взаимосвязей типа "часть-целое" и показывает, из каких элементов состоит система, и как они связаны между собой.

Очевидно, что рассматриваемое в таком аспекте деление системы на составные части представляет собой иерархию, но принципиально отличную от той, которая порождается отношением обобщения. Отличие заключается в том, что части системы никак не обязаны наследовать ее свойства и поведение, поскольку являются самостоятельными сущностями. Более того, части целого обладают собственными атрибутами и операциями, которые существенно отличаются от атрибутов и операций целого. Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой не закрашенный внутри ромб. Этот ромб указывает на тот класс, который представляет собой "целое" или класс-контейнер. Остальные классы являются его "частями" (рис. 22).

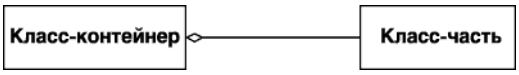


Рис. 22. Графическое изображение отношения агрегации в языке UML

В качестве примера отношения агрегации можно рассмотреть взаимосвязь типа "часть-целое", которая имеет место между классом «Системный блок» персонального компьютера и его составными частями: «Процессор», «Материнская плата», «Оперативная память», «Жесткий диск» и «Дисковод гибких дисков».



Рис. 23. Диаграмма классов для иллюстрации отношения агрегации на примере системного блока ПК

Отношение композиции- частный случай отношения агрегации, и служит для спецификации более сильной формы отношения "часть-целое", при которой составляющие части тесно взаимосвязаны с целым. Особенность: с уничтожением целого уничтожаются и все его составные части.

Композит (composite) - класс, который связан отношением композиции с одним или большим числом классов. Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот класс, который представляет собой класс-композит. Остальные классы являются его "частями" (рис. 24).

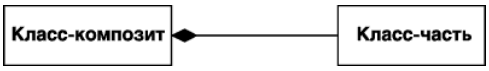


Рис. 24.

Для отношений композиции и агрегации могут использоваться дополнительные обозначения, применяемые для отношения ассоциации. А именно, могут указываться кратности отдельных классов, которые в общем случае не обязательны. Применительно к описанному выше примеру класс Окно программы является классом-композитом, а взаимосвязи составляющих его частей могут быть изображены следующей диаграммой классов (рис. 25).

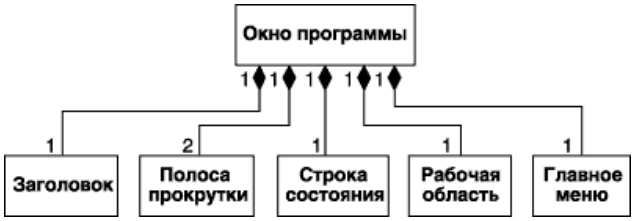


Рис. 25. Диаграмма классов для иллюстрации отношения композиции на примере класса-композита «Окно программы»

Процесс разработки диаграммы классов занимает центральное место при разработке проектов сложных систем. От умения правильно выбрать классы и установить между ними взаимосвязи часто зависит не только успех процесса проектирования, но и производительность выполнения программы. Как показывает практика ООАП, каждый программист в той или иной степени использует личный опыт при разработке новых проектов. Это обусловлено желанием свести новую задачу к уже решенным, чтобы иметь возможность применять не только проверенные фрагменты программного кода, но и отдельные компоненты или библиотеки компонентов.

При определении классов, атрибутов и операций, задании их имен и типов перед разработчиком всегда встает вопрос: какой язык взять за основу.

После разработки диаграммы классов процесс ООАП может быть продолжен в двух направлениях. С одной стороны, если поведение системы тривиально, то можно приступить к разработке диаграмм кооперации и последовательности. Особенности построения диаграмм кооперации будут рассмотрены в следующей главе. Однако для сложных динамических систем, в частности для параллельных систем и систем реального времени, важнейшим аспектом их функционирования является поведение. Специфика поведения таких систем может быть представлена на диаграммах состояний и деятельности, разработка которых в общем случае не обязательна, а определяется конкретным проектом.

Пример: Программное средство представляет собой базу данных «Автоматизация процесса составления расписания в учебном заведении». Программное средство обеспечивает корректировку данных, а именно в БД

«Группы» может изменяться перечень предметов в соответствие с курсом группы и отделением; в БД «Предметы» могут изменяться номера аудиторий и фамилии преподавателей; осуществляет поиск по ФИО преподавателя, номеру аудитории, названию предмета и номеру группы. Программное средство составляет расписание работы для конкретного преподавателя на неделю; для группы на неделю; для группы по конкретному предмету, а также отчет о загрузке аудиторий на каждый день.

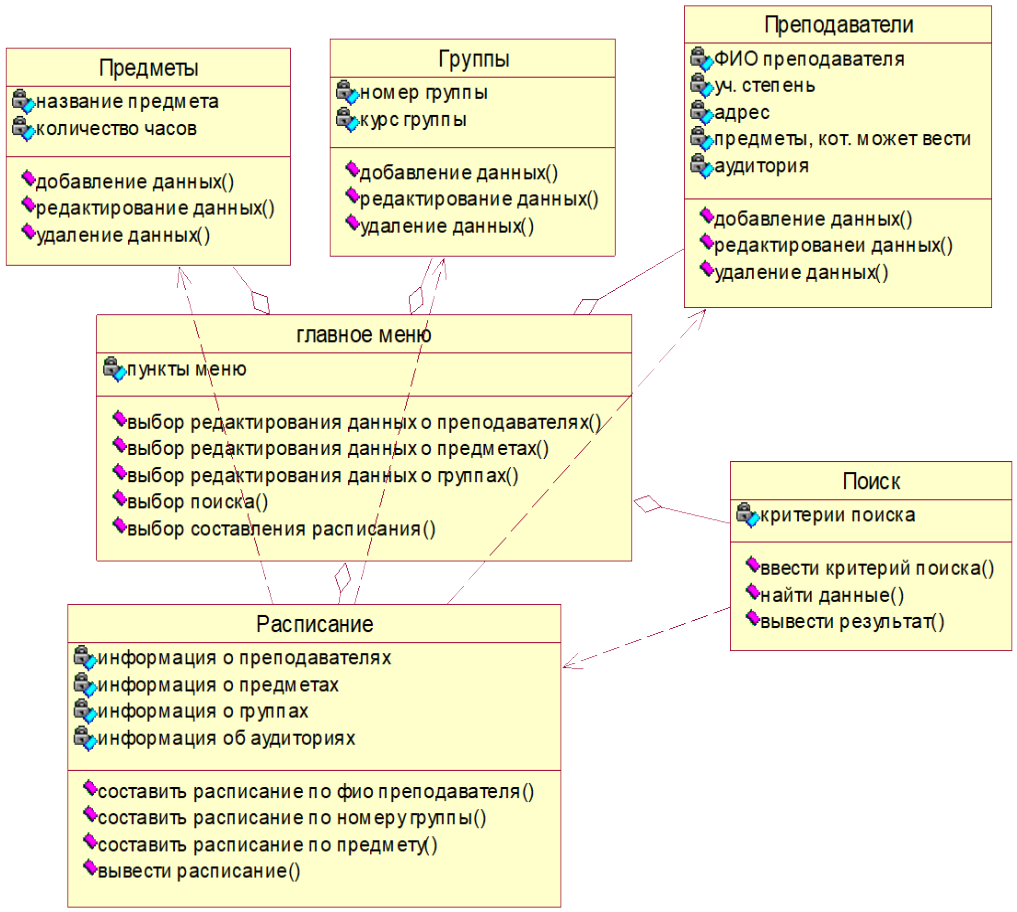


Рис. 26. Пример диаграммы классов

Порядок выполнения работы

1. Изучить теоретические сведения по теме “Диаграмма классов”.
2. Разработать диаграмму классов для произвольной системы индивидуального задания (Приложение А).
3. Оформить отчет, включив в него описание всех компонентов диаграммы классов согласно индивидуальному варианту задания, ответы на контрольные вопросы.

Контрольные вопросы

1. Дайте определение понятию «диаграмма классов» и ее назначение.
2. Дайте определение следующим понятиям «класс», «атрибут», «операция», «отношение», «интерфейс».
3. Опишите отношение ассоциаций между экземплярами классов.
4. Опишите отношение обобщения между экземплярами классов.
5. Опишите отношение агрегации между экземплярами классов.
6. Опишите отношение композиции между экземплярами классов.
7. Приведите пример графического представления основных компонентов диаграммы классов.
8. В чем различие конкретного класса от абстрактного?
9. Дайте определение «квантор видимости» и его классификацию.
10. Какие дополнительные графические примитивы существуют в графической нотации языка UML диаграмме классов?

Критерии оценки:
Всего баллов – 5 (Максимально)
Выполнение заданий лабораторной работы может быть оценено от 0 до 3 баллов.
Ответы на контрольные вопросы – от 0 до 2 баллов.

ЛАБОРАТОРНАЯ РАБОТА № 4.ДИАГРАММА КООПЕРАЦИИ

Цель работы: закрепление теоретических сведений о диаграмме кооперации; овладение практическими навыками моделирования процессов, описывающих взаимодействие объектов в диаграмме кооперации.

Краткие теоретические сведения

Диаграмма кооперации (collaborationdiagram) предназначена для описания поведения системы на уровне отдельных объектов, которые обмениваются между собой сообщениями, чтобы достичь нужной цели или реализовать некоторый вариант использования.

Кооперация(collaboration) — спецификация множества объектов отдельных классов, совместно взаимодействующих с целью реализации отдельных вариантов использования в общем контексте моделируемой системы.

Кооперация определяет структуру поведения системы в терминах взаимодействия участников этой кооперации.

На диаграмме кооперации размещаются объекты, представляющие собой экземпляры классов, связи между ними, которые в свою очередь являются экземплярами ассоциаций, и сообщения. Связи дополняются стрелками сообщений, а также именами ролей, которые играют объекты в данной взаимосвязи. На диаграмме кооперации показываются структурные отношения между объектами в виде различных соединительных линий и изображаются динамические взаимосвязи — потоки сообщений в форме стрелок с указанием направления рядом с соединительными линиями между объектами, при этом задаются имена сообщений и их порядковые номера в общей последовательности сообщений.

Одна и та же совокупность объектов может участвовать в реализации различных кооперaций. В зависимости от рассматриваемой кооперации, могут изменяться как связи между отдельными объектами, так и поток сообщений между ними. Именно это отличает диаграмму кооперации от диаграммы классов, на которой должны быть указаны все без исключения классы, их атрибуты и операции, а также все ассоциации и другие структурные отношения между элементами модели.

Объект(object) — сущность с хорошо определенными границами и индивидуальностью, которая инкапсулирует состояние и поведение.Объект создается на этапе реализации модели или выполнения программы. Он имеет собственное имя и конкретные значения атрибутов.

Для диаграмм кооперации имя объекта – строка текста, разделенная двоеточием: `<собственное имя объекта>'/'<Имя роли класса>:<Имя класса>.`

Имя роли класса указывается в том случае, когда соответствующий класс отсутствует в модели. Имя класса – это имя одного из классов, представленного на диаграмме классов.

Если указано собственное имя объекта, то оно должно начинаться со строчной буквы. Имя объекта, имя роли с символом "/" или имя класса могут отсутствовать, но ":" всегда должно стоять перед именем класса, а "/" – перед именем роли.

Следующие варианты возможных записей полного имени объекта:

- o : C – объект с собственным именем o, экземпляр класса C.
- : C – анонимный объект, экземпляр класса C.
- o : (или o) – объект-сирота с собственным именем o.
- o / R : C – объект с собственным именем o, экземпляр класса C, играющий роль R.
- / R : C – анонимный объект, экземпляр класса C, играющий роль R.
- o / R – объект-сирота с собственным именем o, играющий роль R.
- / R –анонимный объект и одновременно объект-сирота, играющий роль R.



Рис. 27. Примеры графических изображений объектов на диаграммах кооперации

Если собственное имя объекта отсутствует (ставится двоеточие перед именем соответствующего класса (рис. 27, в), то такой объект принято называть *анонимным*. Если отсутствует имя класса – такой объект называется *сиротой*(записывается только собственное имя объекта, двоеточие не ставится, имя класса не указываются (рис. 27, г). Для отдельных объектов (рис. 27, д, е) могут быть дополнительно указаны роли, которые они играют в кооперации.

Все объекты делятся на две категории: пассивные и активные. *Пассивный* объект оперирует только данными и не может инициировать деятельность по управлению другими объектами. Однако пассивные объекты могут посылать сигналы в процессе выполнения запросов, которые они обрабатывают. На диаграмме кооперации пассивные объекты изображаются обычным образом без дополнительных стереотипов.

Активный объект имеет собственный процесс управления и может инициировать деятельность по управлению другими объектами.

Активный объект на диаграмме кооперации обозначается прямоугольником с утолщенными границами (рис. 28). Каждый активный объект является владельцем определенного процесса управления.

Пример: активный объект «а : Клиент» является инициатором открытия счета, который представлен анонимным объектом «: Счет».

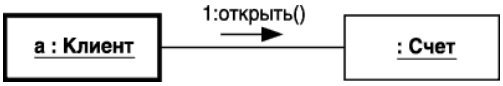


Рис. 28. Графическое изображение активного объекта (слева) на диаграмме кооперации

Мультиобъект (multiobject) представляет собой множество анонимных объектов, которые могут быть образованы на основе одного класса. На диаграмме кооперации мультиобъект используется для того, чтобы показать операции и сигналы, которые адресованы всему множеству анонимных объектов. Мультиобъект изображается двумя прямоугольниками, один из которых выступает из-за верхней правой вершины другого (рис. 29, а). При этом стрелка взаимосвязи относится ко всему множеству объектов, которые обозначает данный мультиобъект. На диаграмме кооперации может быть явно указано отношение агрегации (композиции) между мультиобъектом и отдельным объектом из его множества (рис. 29, б).

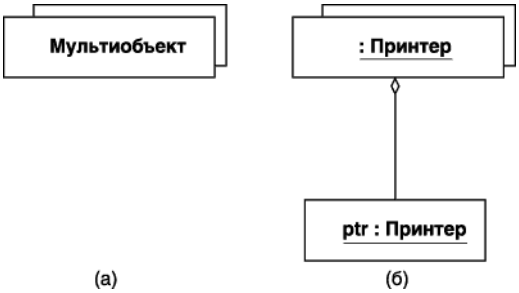


Рис. 29. Графическое изображение мультиобъектов на диаграмме кооперации

Пример: ситуация с отправкой почтового сообщения клиенту из редактора электронной почты (рис. 30). Анонимный активный объект класса «: Редактор Email» вначале посылает сообщение анонимному мультиобъекту класса «Клиент». Это сообщение инициирует выбор единственного объекта класса «Клиент».

После этого выбранному объекту посылается сообщение о необходимости отправить электронное письмо.

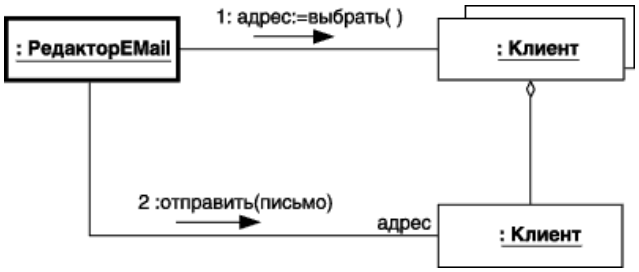


Рис. 30. Фрагмент диаграммы кооперации для выбора адреса клиента для отправки электронного письма

Составной объект (compositeobject) или объект-композит предназначен для представления объекта, имеющего собственную структуру и внутренние потоки управления.

Составной объект является экземпляром класса-композита, который связан отношением композиции со своими частями. На диаграммах кооперации составной объект изображается как обычный объект, состоящий из двух секций: верхней и нижней. В верхней секции записывается имя составного объекта, а в нижней – его объекты-части вместо списка атрибутов (рис. 31).



Рис. 31. Графическое изображение составного объекта на диаграмме кооперации

При изображении диаграммы кооперации отношения между объектами описываются с помощью связей, которые являются экземплярами соответствующих ассоциаций.

При изображении диаграммы кооперации отношения между объектами описываются с помощью связей, которые являются экземплярами соответствующих ассоциаций.

Связь(link) — любое семантическое отношение между некоторой совокупностью объектов.

Бинарная связь изображается отрезком сплошной линии, соединяющей два прямоугольника объектов, на концах линии можно указать имена ролей.

Связи на диаграмме кооперации могут быть только анонимными и при необходимости записываются без двоеточия перед именем ассоциации. Имена связей и кратность концевых точек, как правило, на диаграммах кооперации не указываются. Обозначения агрегации и композиции могут присутствовать на отдельных концах связей.

Пример: обобщенная схема компании с именем «с», которая состоит из департаментов (анонимный мульти-объект класса «Департамент»). В последние входят «Сотрудники». Рефлексивная связь указывает на то, что руководитель департамента является одновременно и его сотрудником.



Рис. 32 Графическое изображение связей с различными стереотипами

Связь может иметь некоторые стереотипы:

- «association» – ассоциация (предполагается по умолчанию, поэтому может не указываться);
- «parameter» – соответствующий объект может быть только параметром метода;
- «local» – область видимости переменной ограничена только соседним объектом;
- «global» – область видимости переменной распространяется на всю диаграмму кооперации;
- «self» – рефлексивная связь объекта с самим собой, которая допускает передачу объектом сообщения самому себе.

Каждое взаимодействие описывается совокупностью сообщений, которыми участвующие в нем объекты обмениваются между собой.

Сообщение(message) — спецификация передачи информации от одного элемента модели к другому с ожиданием выполнения определенных действий со стороны принимающего элемента.

При этом первый объект предполагает, что после получения сообщения вторым объектом последует выполнение некоторого действия. На диаграмме кооперации сообщение является причиной или стимулом начала выполнения операций, отправки сигналов, создания и уничтожения отдельных объектов. Связь обеспечивает канал для направленной передачи сообщений между объектами от объекта-источника к объекту-получателю.

Иногда отправителя сообщения называют клиентом, а получателя – сервером. При этом сообщение от клиента имеет форму запроса некоторого сервиса, а реакция сервера на запрос после получения сообщения может быть связана с выполнением определенных действий или передачи клиенту необходимой информации тоже в форме сообщения.

Сообщения в языке UML специфицируют роли, которые играют объекты – отправитель и получатель сообщения. Сообщения на диаграмме кооперации изображаются дополнительными стрелками рядом с соответствующей связью или ролью ассоциации. Направление стрелки указывает на получателя сообщения. На диаграммах кооперации может использоваться один из трех типов стрелок для обозначения сообщений (рис. 33).

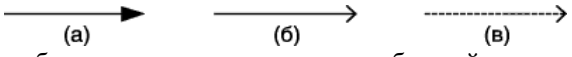


Рис. 33. Графическое изображение различных типов сообщений на диаграмме кооперации

Сплошная линия с треугольной стрелкой (рис. 33, а) обозначает вызов процедуры (операции) или передачу потока управления. Сообщения этого типа могут быть использованы параллельно активными объектами, когда один из них передает сообщение этого типа и ожидает, пока не закончится некоторая последовательность действий, выполняемая вторым объектом. Обычно все такие сообщения синхронны, т. е. инициируются по завершении деятельности или при выполнении определенного условия.

Сплошная линия с V-образной стрелкой (рис. 33, б) обозначает асинхронное сообщение в простом потоке управления. В этом случае клиент передает асинхронное сообщение и продолжает выполнять свою деятельность, не ожидая ответа от сервера.

Пунктирная линия с V-образной стрелкой (рис. 33, в) обозначает возврат из вызова процедуры. Стрелки этого типа зачастую отсутствуют на диаграммах кооперации, поскольку неявно предполагается их существование после окончания процесса выполнения операции или деятельности.

Каждое сообщение может быть помечено строкой текста, которая имеет следующий формат:

<Предшествующие сообщения><Выражение последовательности><Возвращаемое значе-

ние := имя сообщения>(Список аргументов)

Предшествующие сообщения — это разделенные запятыми номера сообщений, записанные перед наклонной чертой: <Номер сообщения','>* <'/'>. Если список номеров сообщений пуст, то вся запись, включая наклонную черту, опускается. Если номера сообщений указываются, то они должны соответствовать номерам других сообщений на этой же диаграмме кооперации. Смысл указания предшествующих сообщений заключается в том, что данное сообщение не может быть передано, пока не будут переданы своим адресатам все сообщения, номера которых записаны в данном списке.

Выражение последовательности — это разделенный точками список отдельных термов последовательностей, после которого записывается двоеточие: <Терм последовательности'.'…> ':'

Каждый из термов представляет отдельный уровень процедурной вложенности в форме законченной итерации. Наиболее верхний уровень соответствует самому левому терму последовательности. Если все потоки управления параллельные, то вложенность отсутствует. Каждый терм последовательности имеет следующий синтаксис: [Целое число | Имя] [Рекуррентность].

Целое число указывает на порядковый номер сообщения в процедурной последовательности верхнего уровня. Сообщения, номера которых отличаются на единицу, следуют подряд один за другим.

Имя в форме буквы алфавита используется для спецификации параллельных потоков (нитей) управления. Сообщения, которые отличаются только именем, являются параллельными на этом уровне вложенности. На одном уровне вложенности все нити управления эквивалентны в смысле приоритета передачи сообщений.

Рекуррентность используется для указания итеративного или условного характера выполнения передачи сообщений. Семантика рекуррентности представляет ноль или больше сообщений, которые должны быть выполнены в зависимости от записанного условия. Возможны два варианта записи рекуррентности:

- '*'['Предложение-итерация']' для записи итеративного выполнения соответствующего выражения. Итерация представляет последовательность сообщений одного уровня вложенности. Предложение-итерация может быть опущено, если количество итераций никак не специфицируется. Наиболее часто предложение-итерация записывается на псевдокоде или языке программирования. В языке UML формат записи этого предложения строгим образом не определен.
- '['Предложение-условие']' для записи ветвления. Эта форма записи специфицирует условие для данного сообщения, передача которого по данной ветви возможна только при его истинности.

В общем случае предложение-условие - обычное булевское выражение и предназначено для синхронизации отдельных нитей потока управления. Записывается в квадратных скобках и может быть опущено, если оно отсутствует у данного сообщения. Наличие этого условия обеспечивает передачу сообщения только в том случае, если это условие принимает значение "истина". Предложение-условие может быть записано на обычном тексте, псевдокоде или некотором языке программирования.

Предложение-условие записывается так же, как и итерация, но без звездочки. Это можно понимать как некоторую одношаговую итерацию. В общем случае предполагается, что специфицированная итерация выполняется последовательно. Если необходимо отметить возможность параллельного выполнения итерации, то для этой цели в языке UML используется символ "*||". Итерация не распространяется на вложенные уровни данного потока или нити. Каждый уровень должен иметь собственное представление для итеративного повторения процедурной последовательности.

Имя сообщения, записанное в сигнатуре после возвращаемого значения, означает имя события, которое инициируется объектом-получателем сообщения после его приема. Наиболее часто таким событием является вызов операции у объекта-получателя. Это может быть реализовано различными способами, один из которых – явное указание в качестве имени сообщения вызываемой операции. Тогда соответствующая операция должна быть определена в том классе, которому принадлежит объект-получатель.

Список аргументов представляет собой разделенные запятыми и заключенные в круглые скобки действительные параметры той операции, вызов которой инициируется данным сообщением. Список аргументов может быть пустым, однако скобки все равно записываются. Для записи аргументов также может быть использован некоторый псевдокод или язык программирования.

В языке UML предусмотрены стандартные действия, выполняемые в ответ на получение соответствующего сообщения. Они могут быть явно указаны на диаграмме кооперации в форме стереотипа перед именем сообщения, к которому они относятся, или выше его.

В языке UML определены следующие стереотипы сообщений:

- «call» (вызвать) – сообщение, требующее вызова операции или процедуры объекта-получателя. Если сообщение с этим стереотипом рефлексивное, то оно инициирует локальный вызов операции у пославшего это сообщение объекта.
- «return»(возвратить) – сообщение, возвращающее значение выполненной операции или процедуры вызвавшему ее объекту. Значение результата может инициировать ветвление потока управления.
- «create»(создать) – сообщение, требующее создания другого объекта для выполнения определенных

действий. Созданный объект может стать активным (ему передается поток управления), а может остаться пассивным.

− «destroy»(уничтожить) – сообщение с явным требованием уничтожить соответствующий объект. Посылается в том случае, когда необходимо прекратить нежелательные действия со стороны существующего в системе объекта, либо когда объект больше не нужен и должен освободить задействованные им системные ресурсы.

− «send»(послать) – обозначает посылку другому объекту сигнала, который асинхронно инициируется одним объектом и принимается (перехватывается) другим. Отличие сигнала от сообщения заключается в том, что сигнал должен быть явно описан в том классе, объект которого инициирует его передачу.

Построение диаграммы кооперации можно начинать сразу после построения диаграммы классов. При разработке диаграмм кооперации вначале изображаются объекты и связи между ними. Далее на диаграмму кооперации необходимо нанести все сообщения, указав их порядок и другие семантические особенности. Эта диаграмма может содержать только те объекты и связи, которые уже определены на построенной ранее диаграмме классов.

Процесс построения диаграммы кооперации должен быть согласован с процессами построения диаграммы классов и диаграммы последовательности. В первом случае, как уже отмечалось, необходимо следить за использованием только тех объектов, для которых определены порождающие их классы. Во втором случае необходимо согласовывать последовательности передаваемых сообщений. Речь идет о том, что не допускается различный порядок следования сообщений для моделирования одного и того же взаимодействия на диаграмме кооперации и диаграмме последовательности.

Диаграмма кооперации, с одной стороны, обеспечивает концептуально согласованный переход от статической модели диаграммы классов к динамическим моделям поведения, представляемым диаграммами последовательности, состояний и деятельности. С другой стороны, диаграмма этого типа предопределяет особенности реализации модели на диаграммах компонентов и развертывания.

Пример: Программное средство представляет среду для формирования отчетов по лекционному материалу. Пользователь может вводить свою информацию в отчеты, изменять параметры. После оформления отчетов пользователю предоставлена возможность сохранение отчета.



Рис. 34. Пример диаграммы кооперации

Порядок выполнения работы

1. Изучить теоретические сведения по теме “Диаграмма кооперации”.
2. Разработать диаграмму кооперации для произвольной системы индивидуального задания (Приложение А).
3. Оформить отчет, включив в него описание всех компонентов диаграммы кооперации согласно

индивидуальному варианту задания, ответы на контрольные вопросы.

Контрольные вопросы

1. Дайте определение понятию «диаграмма кооперации».
2. Опишите назначение диаграммы кооперации.
3. Что из себя представляет объект в диаграмме кооперации? Как полностью записывается имя объекта? Назовите варианты записи объекта.
4. На какие две категории делятся объекты?
5. Дайте определение понятиям «мультиобъект», «составной объект». Каково их графическое изображение?
6. Что понимается под связью в диаграмме кооперации? Какие типы связи возможны между объектами на диаграмме кооперации?
7. Дайте определение понятию «сообщение» в диаграмме кооперации и назовите его формат записи. Графическое изображение сообщений.
8. Какие стереотипы сообщений существуют в языке UML? Охарактеризуйте каждый из них.

Критерии оценки:
Всего баллов – 5 (Максимально)
Выполнение заданий лабораторной работы может быть оценено от 0 до 3 баллов.
Ответы на контрольные вопросы – от 0 до 2 баллов.

ЛАБОРАТОРНАЯ РАБОТА № 5. ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ

Цель работы: закрепление теоретических сведений о диаграмме последовательности; овладение практическими навыками моделирования процессов, описывающих взаимодействие объектов в диаграмме последовательности.

Краткие теоретические сведения

Диаграмма последовательности (sequencediagram) – диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления.

На диаграмме последовательности неявно присутствует ось времени, что позволяет визуализировать временные отношения между передаваемыми сообщениями. С помощью диаграммы последовательности можно представить взаимодействие элементов модели как своеобразный временной график "жизни" всей совокупности объектов, связанных между собой для реализации варианта использования программной системы, достижения цели или выполнения задачи.

На диаграмме последовательности изображаются объекты, которые непосредственно участвуют во взаимодействии, при этом никакие статические связи с другими объектами не визуализируются. Диаграмма последовательности имеет как бы два измерения: одно - слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта; второе – вертикальная временная ось, направленная сверху вниз.

Каждый объект графически изображается в форме прямоугольника и располагается в верхней части своей линии жизни (рис. 35). Внутри прямоугольника записываются собственное имя объекта и имя класса, разделенные двоеточием, запись подчеркивается, что является признаком объекта, который, как указывалось ранее, представляет собой экземпляр класса.

Объекты записываются так же, как и в диаграмме кооперации. **Анонимный объект** – такой объект, для которого отсутствует собственное имя, но указано имя класса. **Объект-сирота** – такой объект, для которого может отсутствовать имя класса, но указано собственное имя объекта. Роль классов в именах объектов на диаграммах последовательности, как правило, не указывается.

Крайним слева на диаграмме изображается объект - инициатор моделируемого процесса взаимодействия (объект а на рис. 35). Правее - другой объект, который непосредственно взаимодействует с первым, т.е. порядок расположения объектов на диаграмме последовательности определяется исключительно соображениями удобства визуализации их взаимодействия друг с другом.

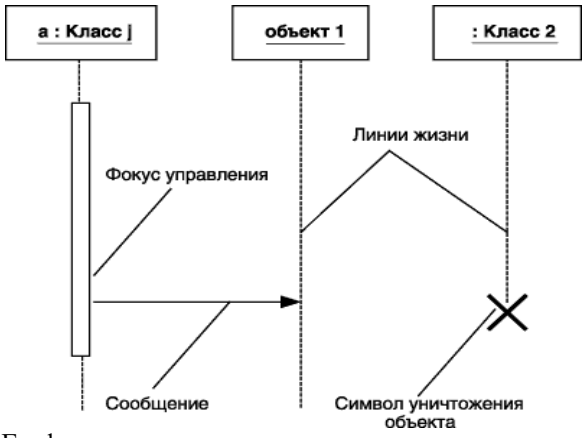


Рис. 35. Графические элементы диаграммы последовательности

Начальному моменту времени соответствует самая верхняя часть диаграммы. При этом процесс взаимодействия объектов реализуется посредством сообщений, которые посылаются одними объектами другим. Сообщения изображаются в виде горизонтальных стрелок с именем сообщения и образуют определенный порядок относительно времени своей инициализации. Другими словами, сообщения, расположенные на диаграмме последовательности выше, передаются раньше тех, которые расположены ниже. При этом масштаб на оси времени не указывается, поскольку диаграмма последовательности моделирует лишь временную упорядоченность взаимодействий типа "раньше-позже".

Линия жизни объекта (objectlifeline) – вертикальная линия на диаграмме последовательности, которая представляет существование объекта в течение определенного периода времени. Линия жизни объекта изображается пунктирной вертикальной линией, ассоциированной с единственным объектом на диаграмме последовательности. Линия жизни служит для обозначения периода времени, в течение которого объект существует в системе и может потенциально участвовать во всех ее взаимодействиях. Если объект существует в системе постоянно, то и его линия жизни должна продолжаться по всей рабочей области диаграммы последовательности от самой верхней ее части до самой нижней («объект 1» и анонимный объект «Класса 2»).

Отдельные объекты, закончив выполнение своих операций, могут быть уничтожены, чтобы освободить занимаемые ими ресурсы. Для таких объектов линия жизни обрывается в момент его уничтожения. Для обозначения момента уничтожения объекта в языке UML применяется специальный символ в форме латинской буквы "X". На рис. 36 этот символ используется для уничтожения анонимного объекта, образованного от «Класса 3» . Ниже этого символа пунктирная линия не изображается, поскольку соответствующего объекта в системе уже нет, и этот объект должен быть исключен из всех последующих взаимодействий.

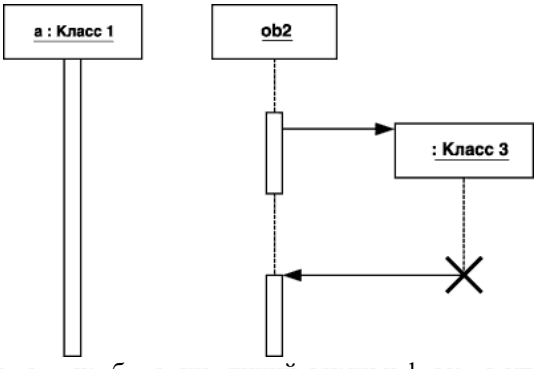


Рис. 36. Графическое изображение линий жизни и фокусов управления объектов

Отдельные объекты в системе могут создаваться по мере необходимости, существенно экономя ресурсы системы и повышая ее производительность. В этом случае прямоугольник такого объекта изображается не в верхней части диаграммы последовательности, а в той, которая соответствует моменту создания объекта (анонимный объект, образованный от «Класса 3» на рис. 36). При этом прямоугольник объекта вертикально располагается в том месте диаграммы, которое по оси времени совпадает с моментом его возникновения в системе. Объект создается со своей линией жизни а, возможно, и с фокусом управления.

В процессе функционирования объектно-ориентированных систем одни объекты могут находиться в активном состоянии, непосредственно выполняя определенные действия, или в состоянии пассивного

ожидания сообщений от других объектов. Фокус управления - символ, применяемый для того, чтобы явно выделить подобную активность объектов на диаграммах последовательности.

Фокус управления (focusofcontrol) - специальный символ на диаграмме последовательности, указывающий период времени, в течение которого объект выполняет некоторое действие, находясь в активном состоянии.

Фокус управления изображается в форме вытянутого узкого прямоугольника («объект а» на рис. 35), верхняя сторона которого обозначает начало получения фокуса управления объекта (начало активности), а ее нижняя сторона - окончание фокуса управления (окончание активности). Этот прямоугольник располагается ниже обозначения соответствующего объекта и может заменять его линию жизни («объект а» на рис. 36), если на всем ее протяжении он активен.

Периоды активности объекта могут чередоваться с периодами его пассивности или ожидания. В этом случае у такого объекта фокусы управления изменяют свое изображение на линию жизни и наоборот (объект сирота «ob2» на рис. 36). Получить фокус управления может только объект, у которого в этот момент имеется линия жизни. Если же объект был уничтожен, то вновь возникнуть в системе он уже не может. Вместо него может быть создан лишь экземпляр этого же класса, который, строго говоря, будет другим объектом.

В отдельных случаях объект может посылать сообщения самому себе, инициируя так называемые *рефлексивные* сообщения. Для этой цели служит специальное изображение (сообщение у «объекта а» на рис. 37). Такие сообщения изображаются в форме сообщения, начало и конец которого соприкасаются с линией жизни или фокусом управления одного и того же объекта. Подобные ситуации возникают, например, при обработке нажатий на клавиши клавиатуры при вводе текста в редактируемый документ, при наборе цифр номера телефона абонента.

Если в результате рефлексивного сообщения создается новый подпроцесс или нить управления, то говорят о рекурсивном или вложенном фокусе управления. На диаграмме последовательности рекурсия обозначается небольшим прямоугольником, присоединенным к правой стороне фокуса управления того объекта, для которого изображается данное рекурсивное взаимодействие (анонимный объект «Класса 2» на рис. 37).

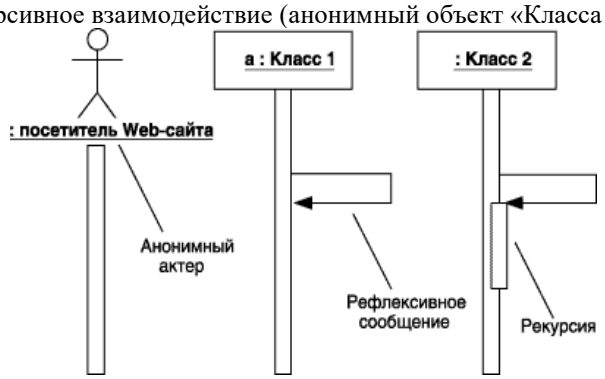


Рис. 37. Графическое изображение актера, рефлексивного сообщения и рекурсии

Сообщения на диаграмме последовательности аналогичны сообщениям на диаграмме кооперации, но имеют дополнительные семантические особенности. На диаграмме последовательности все сообщения упорядочены по времени своей передачи в моделируемой системе.

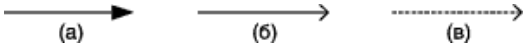


Рис. 38. Графическое изображение различных видов сообщений между объектами на диаграмме последовательности

Первая разновидность сообщения (рис. 38, а) наиболее распространена и используется для вызова процедур, выполнения операций или обозначения отдельных вложенных потоков управления. Начало этой стрелки, как правило, соприкасается с фокусом управления того объекта-клиента, который инициирует это сообщение. Конец стрелки соприкасается с линией жизни того объекта, который принимает это сообщение и выполняет в ответ определенные действия. При этом принимающий объект может получить фокус управления, становясь в этом случае активным. Передающий объект может потерять фокус управления или остаться активным.

Вторая разновидность сообщения (рис. 38, б) используется для обозначения простого асинхронного сообщения, которое передается в произвольный момент времени. Передача такого сообщения обычно не сопровождается получением фокуса управления объектом-получателем.

Третья разновидность сообщения (рис. 38, в) используется для возврата из вызова процедуры. Примером может служить простое сообщение о завершении вычислений без предоставления результата расчетов объекту-

клиенту. В процедурных потоках управления эта стрелка может быть опущена, поскольку ее наличие неявно предполагается в конце активизации объекта. В то же время считается, что каждый вызов процедуры имеет свою пару - возврат вызова. Для непроцедурных потоков управления, включая параллельные и асинхронные сообщения, стрелка возврата должна указываться явным образом.

Обычно сообщения изображаются горизонтальными стрелками, соединяющими линии жизни или фокусы управления двух объектов на диаграмме последовательности. При этом неявно предполагается, что время передачи сообщения достаточно мало по сравнению с процессами выполнения действий объектами. Считается также, что за время передачи сообщения с соответствующими объектами не может произойти никаких событий. Другими словами, состояния объектов не изменяются. Если же это предположение не может быть признано справедливым, то стрелка сообщения изображается под наклоном, так чтобы конец стрелки располагался ниже ее начала.

Каждое сообщение на диаграмме последовательности ассоциируется с определенной операцией, которая должна быть выполнена принявшим его объектом. При этом операция может иметь аргументы или параметры, значения которых влияют на получение различных результатов. Соответствующие параметры операции будет иметь и вызывающее это действие сообщение. Более того, значения параметров отдельных сообщений могут содержать условные выражения, образуя ветвление или альтернативные пути основного потока управления.

Одна из особенностей диаграммы последовательности - возможность визуализировать простое *ветвление процесса.* Для изображения ветвления используются две или более стрелки, выходящие из одной точки фокуса управления объекта (объект «ob1» на рис. 39). При этом рядом с каждой из них должно быть явно указано соответствующее условие ветви в форме булевского выражения.

Количество ветвей может быть произвольным, но наличие ветвлений может существенно усложнить интерпретацию диаграммы последовательности. Предложение-условие должно быть явно указано для каждой ветви и записывается в форме обычного текста, псевдокода или выражения языка программирования. Это выражение всегда должно возвращать некоторое булевское выражение. Запись этих условий должна исключать одновременную передачу альтернативных сообщений по двум и более ветвям. В противном случае на диаграмме последовательности может возникнуть конфликт ветвления.

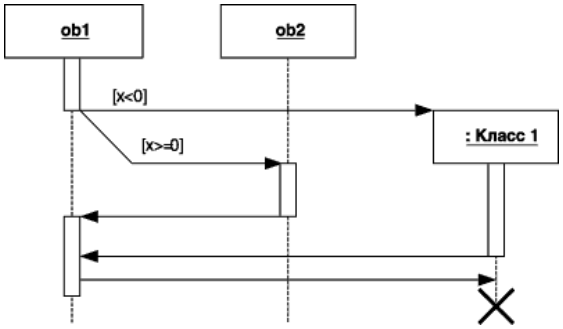


Рис. 6.5. Графическое изображение бинарного ветвления потока управления на диаграмме последовательности

С помощью ветвления можно изобразить и более сложную логику взаимодействия объектов между собой (объект «ob1» на рис.40). Если условий более двух, то для каждого из них необходимо предусмотреть ситуацию единственного выполнения. Описанный ниже пример относится к моделированию взаимодействия программной системы обслуживания клиентов в банке.

Пример: объект «ob1» вызывает выполнение действий у одного из трех других объектов. Условием ветвления может служить сумма снимаемых клиентом средств со своего текущего счета. Если эта сумма превышает 1500$, то могут потребоваться дополнительные действия, связанные с созданием и последующим разрушением объекта «Класса 1». Если же сумма превышает 100$, но не превышает 1500$, то вызывается операция или процедура объекта «ob3». И, наконец, если сумма не превышает 100$, то вызывается операция или процедура объекта «ob2». При этом объекты «ob1», «ob2» и «ob3» постоянно существуют в системе. Последний объект создается от «Класса 1» только в том случае, если справедливо первое из альтернативных условий. В противном случае он может быть никогда не создан.

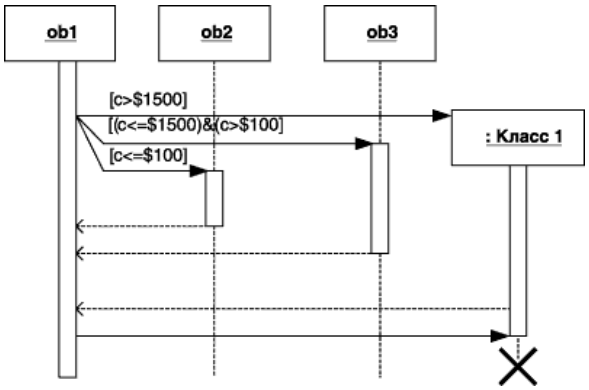


Рис. 40. Графическое изображение тернарного ветвления потока управления на диаграмме последовательности

Объект «ob1» имеет постоянный фокус управления, а все остальные объекты - получают фокус управления только для выполнения ими соответствующих операций. На диаграммах последовательности при записи сообщений также могут использоваться стереотипы. Ниже представлена диаграмма последовательности для описанного выше случая ветвления, дополненная стереотипными значениями отдельных сообщений (рис. 41).

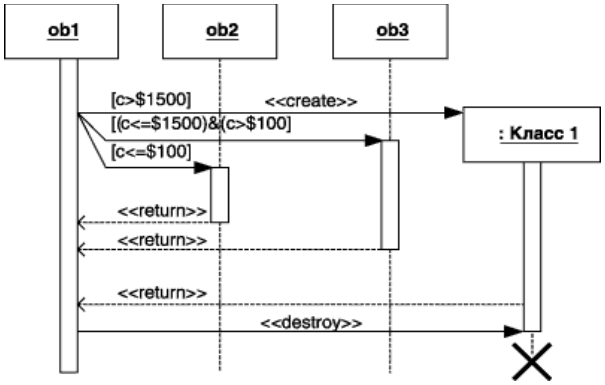


Рис. 41. Диаграмма последовательности со стереотипными значениями сообщений

Сообщения могут иметь собственное имя, в качестве которого выступает имя операции, вызов которой инициируют эти сообщения у принимающего объекта. Тогда со стрелкой записывается имя операции с круглыми скобками, в которых могут указываться параметры или аргументы соответствующей операции. Если параметры отсутствуют, то скобки все равно изображаются. Построение диаграммы последовательности начинают с выделения из всей совокупности классов только тех, объекты которых участвуют во взаимодействии. После все объекты наносятся на диаграмму, с соблюдением порядка инициализации сообщений.

Когда объекты визуализированы, можно приступать к спецификации сообщений, учитывая те операции, которые имеют классы соответствующих объектов, иногда используя их стереотипы.

Для уничтожения объектов, которые создаются на время выполнения своих действий, нужно предусмотреть явное сообщение. Наиболее простые случаи ветвления процесса взаимодействия можно изобразить на одной диаграмме, а в более сложных случаях для моделирования каждой ветви управления может потребоваться отдельная диаграмма последовательности. Общим правилом является визуализация особенностей реализации каждого варианта использования на отдельной диаграмме последовательности.

Пример: Программное средство представляет среду для формирования отчетов по лекционному материалу. Пользователь может вводить свою информацию в отчеты, изменять параметры. После оформления отчетов пользователю предоставлена возможность сохранение отчета. Вывода его на печать и его архивацию.

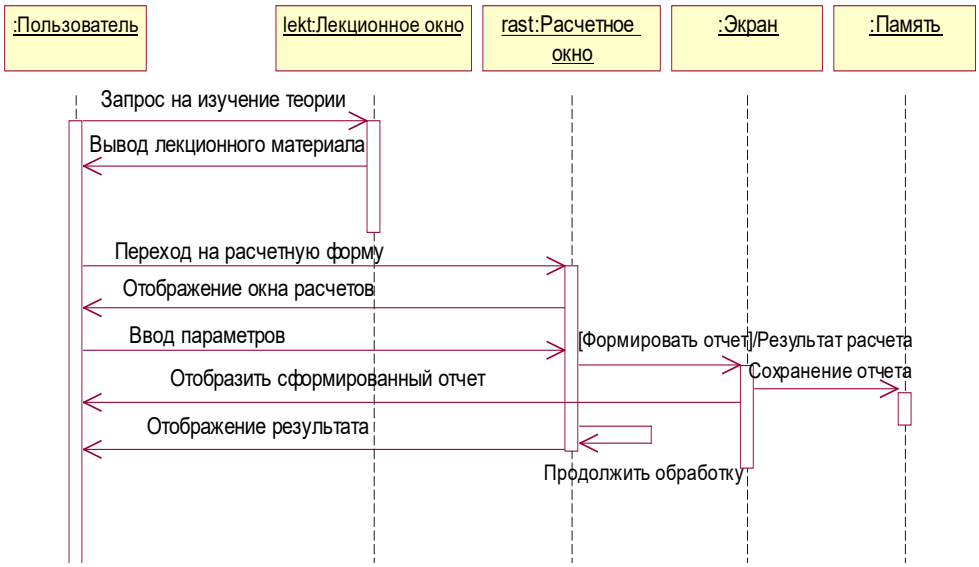


Рис. 42. Пример диаграммы последовательности

Порядок выполнения работы

1. Изучить теоретические сведения по теме “Диаграмма последовательности”.
2. Разработать диаграмму последовательности для произвольной системы индивидуального задания (Приложение А).
3. Оформить отчет, включив в него описание всех компонентов диаграммы последовательностисогласно индивидуальному варианту задания, ответы на контрольные вопросы.

Контрольные вопросы

1. Дайте определение понятию «диаграмма последовательности».
2. Опишите назначение диаграммы последовательности.
3. В чем различие анонимного объекта и объекта-сироты?
4. Охарактеризуйте линию жизни объекта. Ее графическое изображение.
5. Для чего предназначен фокус управления? Его графическое изображение.
6. Дайте определение понятию «сообщение». Какие виды сообщений используются в диаграмме последовательности? Когда используется рефлексивное сообщение?
7. Расскажите про ветвления на диаграмме последовательности. Приведите пример ветвления.

Критерии оценки:
Всего баллов – 5 (Максимально)
Выполнение заданий лабораторной работы может быть оценено от 0 до 3 баллов.
Ответы на контрольные вопросы – от 0 до 2 баллов.

ЛАБОРАТОРНАЯ РАБОТА № 6. ДИАГРАММА СОСТОЯНИЙ

Цель работы: закрепление теоретических сведений о диаграмме состояний; овладение практическими навыками моделирования процессов, описывающих взаимодействие объектов в диаграмме состояний.

Краткие теоретические сведения

Для представления динамических особенностей взаимодействия элементов модели, в контексте реализации вариантов использования, предназначены диаграммы кооперации и последовательности. Однако для моделирования процессов функционирования большинства сложных систем, особенно систем реального времени, этих представлений недостаточно.

Диаграмма состояний (statechartdiagram) - диаграмма, которая представляет конечный автомат, в которой вершины обозначают состояния, а дуги показывают переходы между двумя состояниями.

Назначение диаграммы состояний - описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение моделируемой системы в течение всего ее жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий. Системы, которые реагируют на внешние действия от других систем или от пользователей, иногда называют *реактивными*. Если такие действия инициируются в произвольные случайные моменты времени, то говорят об асинхронном поведении модели.

Диаграммы состояний используются для описания поведения отдельных систем и подсистем. Они также могут быть применены для спецификации функциональности экземпляров отдельных классов, т. е. для моделирования всех возможных изменений состояний конкретных объектов. Диаграмма состояний по существу является графом специального вида, который служит для представления конечного автомата.

Диаграммы состояний могут быть вложены друг в друга, образуя вложенные диаграммы для более детального представления состояний отдельных элементов модели. Для понимания семантики конкретной диаграммы состояний необходимо представлять особенности поведения моделируемой сущности, а также иметь общие сведения из теории конечных автоматов.

Конечный автомат (statemachine) - модель для спецификации поведения объекта в форме последовательности его состояний, которые описывают реакцию объекта на внешние события, выполнение объектом действий, а также изменение его отдельных свойств.

В контексте языка UML понятие конечного автомата обладает дополнительной семантикой. Вершинами графа конечного автомата являются состояния и другие типы элементов модели, которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Конечный автомат описывает поведение отдельного объекта в форме последовательности состояний, охватывающих все этапы его жизненного цикла, начиная от создания объекта и заканчивая его уничтожением. Каждая диаграмма состояний представляет собой конечный автомат.

Основными понятиями, характеризующими конечный автомат, являются состояние и переход. Различие между ними заключается в том, что длительность нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое. Другими словами, переход объекта из состояния в состояние происходит мгновенно.

В общем случае конечный автомат представляет динамические аспекты моделируемой системы в виде ориентированного графа, вершины которого соответствуют состояниям, а дуги - переходам. При этом поведение моделируется, как последовательное перемещение по графу состояний от вершины к вершине по связывающим их дугам с учетом их ориентации. Для графа состояний системы можно ввести в рассмотрение специальные свойства.

Среди таких свойств - выделение из всей совокупности состояний двух специальных: начального и конечного. Ни в графе состояний, ни на диаграмме состояний время нахождения системы в том или ином состоянии явно не учитывается, однако предполагается, что последовательность изменения состояний упорядочена во времени. Другими словами, каждое последующее состояние может наступить позже предшествующего ему состояния.

Состояние (state) - условие или ситуация в ходе жизненного цикла объекта, в течение которого он удовлетворяет логическому условию, выполняет определенную деятельность или ожидает события.

Состояние может быть задано в виде набора конкретных значений атрибутов объекта некоторого класса, при этом изменение отдельных значений этих атрибутов будет отражать изменение состояния моделируемого объекта или системы в целом. Однако не каждый атрибут класса может характеризовать состояние его объектов. Как правило, имеют значение только те свойства элементов системы, которые отражают динамический или функциональный аспект ее поведения. В этом случае состояние будет характеризоваться некоторым инвариантным условием, включающим в себя только принципиальные для поведения объекта или системы атрибуты классов и их значения.

Такое условие может соответствовать ситуации, когда моделируемый объект находится в состоянии ожидания возникновения внешнего события. В то же время нахождение объекта в некотором состоянии может быть связано с выполнением определенных действий. В последнем случае соответствующая деятельность начинается в момент перехода моделируемого элемента в рассматриваемое состояние, а после и элемент может покинуть данное состояние в момент завершения этой деятельности.

Состояние на диаграмме изображается прямоугольником со скругленными вершинами (рис. 43). Этот прямоугольник может быть разделен на две секции. Если указана лишь одна секция, то в ней записывается только имя состояния (рис. 43, а). В противном случае в первой из них записывается имя состояния, а во второй - список некоторых внутренних действий или переходов в данном состоянии (рис. 43, б). При этом под действием в языке UML понимают некоторую атомарную операцию, выполнение которой приводит к изменению состояния или возврату некоторого значения (например, "истина" или "ложь").

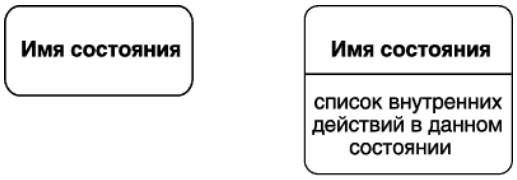


Рис. 43. Графическое изображение состояний на диаграмме состояний

Имя у состояния может отсутствовать, т. е. оно необязательно для некоторых состояний. В этом случае состояние является *анонимным*. Если на одной диаграмме состояний несколько анонимных состояний, то все они должны различаться между собой.

Действие (action) - спецификация выполнимого утверждения, которая образует абстракцию вычислительной процедуры.

Действие обычно приводит к изменению состояния системы, и может быть реализовано посредством передачи сообщения объекту, модификацией связи или значения атрибута. Для ряда состояний может потребоваться дополнительно указать действия, которые должны быть выполнены моделируемым элементом. Для этой цели служит добавочная секция в обозначении состояния, содержащая перечень внутренних действий или деятельность, которые производятся в процессе нахождения моделируемого элемента в данном состоянии. Каждое действие записывается в виде отдельной строки и имеет следующий формат:

```
<метка действия '/ ' выражение действия>
```

Метка действия указывает на обстоятельства или условия, при которых будет выполняться деятельность, определенная выражением действия. При этом выражение действия может использовать любые атрибуты и связи, принадлежащие области имен или контексту моделируемого объекта. Если список выражений действия пустой, то метка действия с разделителем в виде наклонной черты '/' не указывается. Перечень меток действий в языке UML фиксирован, причем эти метки не могут быть использованы в качестве имен событий:

Входное действие (entryaction) - действие, которое выполняется в момент перехода в данное состояние. Обозначается с помощью ключевого слова - метки действия entry, которое указывает на то, что следующее за ней выражение действия должно быть выполнено в момент входа в данное состояние.

Действие выхода(exitaction) - действие, производимое при выходе из данного состояния. Обозначается с помощью ключевого слова - метки действия exit, которое указывает на то, что следующее за ней выражение действия должно быть выполнено в момент выхода из данного состояния.

Внутренняя деятельность(doactivity) - выполнение объектом операций или процедур, которые требуют определенного времени. Обозначается с помощью ключевого слова - метки деятельности do, которое специфицирует так называемую "ду-деятельность", выполняемую в течение всего времени, пока объект находится в данном состоянии, или до тех пор, пока не будет прервано внешним событием. При нормальном завершении внутренней деятельности генерируется соответствующее событие. Во всех остальных случаях метка действия идентифицирует событие, которое запускает соответствующее выражение действия. Эти события называются внутренними переходами. Семантически они эквивалентны переходам в само это состояние, за исключением той особенности, что выход из этого состояния или повторный вход в него не происходит. Это означает, что действия входа и выхода не производятся. При этом выполнение внутренних действий в состоянии не может быть прервано никакими внешними событиями, в отличие от внутренней деятельности, выполнение которой требует определенного времени.

Пример: аутентификацию клиента для доступа к ресурсам моделируемой информационной системы (рис. 44). Список внутренних действий в данном состоянии может включать следующие действия. Первое действие - входное, которое выполняется при входе в это состояние и связано с получением строки символов, соответствующих паролю клиента. Далее выполняется деятельность по проверке введенного клиентом пароля. При успешном завершении этой проверки выполняется действие на выходе, которое отображает меню доступных для клиента опций.

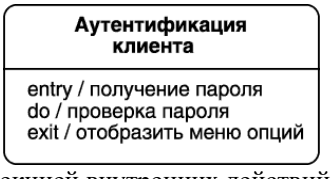


Рис. 44. Пример состояния с непустой секцией внутренних действий

Кроме обычных состояний на диаграмме состояний могут размещаться псевдосостояния.

Псевдосостояние (pseudo-state) - вершина в конечном автомате, которая имеет форму состояния, но не обладает поведением.

Примерами псевдосостояний, которые определены в языке UML, являются начальное и конечное состояния.

Начальное состояние(startstate) - разновидность псевдосостояния, обозначающее начало выполнения процесса изменения состояний конечного автомата или нахождения моделируемого объекта в составном состоянии.

В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания на диаграмме состояний графической области, от которой начинается процесс изменения состояний. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка (рис. 45, а), из которого может только выходить стрелка-переход.



Рис. 45. Графическое изображение начального и конечного состояний

На самом верхнем уровне представления объекта переход из начального состояния может быть помечен событием создания (инициализации) данного объекта. В противном случае этот переход никак не помечается. Если этот переход не помечен, то он является первым переходом на диаграмме состояний в следующее за ним состояние. Каждая диаграмма или под-диаграмма состояний должна иметь единственное начальное состояние.

Конечное состояние(finalstate)- разновидность псевдосостояния, обозначающее прекращение процесса изменения состояний конечного автомата или нахождения моделируемого объекта в составном состоянии.

В этом состоянии должен находиться моделируемый объект или система по умолчанию после завершения работы конечного автомата. Оно служит для указания на диаграмме состояний графической области, в которой завершается процесс изменения состояний или жизненный цикл данного объекта.

Графически конечное состояние в языке UML обозначается в виде закрашенного кружка, помещенного в окружность (рис. 45, б), в которую может только входить стрелка-переход. Каждая диаграмма состояний или подсостояний может иметь несколько конечных состояний, при этом все они считаются эквивалентными на одном уровне вложенности состояний.

Пребывание объекта в первом состоянии может сопровождаться выполнением некоторых внутренних действий или деятельности. При этом изменение текущего состояния объекта будет возможно либо после завершения этих действий, либо при возникновении некоторых внешних событий. В обоих случаях говорят, что происходит переход объекта из одного состояния в другое.

Переход (transition) - отношение между двумя состояниями, которое указывает на то, что объект в первом состоянии должен выполнить определенные действия и перейти во второе состояние.

Переход осуществляется при наступлении некоторого события: окончания выполнения деятельности (doactivity), получении объектом сообщения или приемом сигнала. На переходе указывается имя события, а также действия, производимые объектом в ответ на внешние события при переходе из одного состояния в другое.

Переход может быть направлен в то же состояние, из которого он выходит. В этом случае его называют переходом в себя. Исходное и целевое состояния перехода в себя совпадают. Этот переход изображается петлей со стрелкой и отличается от внутреннего перехода. При переходе в себя объект покидает исходное состояние, а затем снова входит в него. При этом всякий раз выполняются внутренние действия, специфицированные метками entry и exit.

Срабатывание <*перехода*>(fire) - выполнение перехода из одного состояния в другое.

Срабатывание перехода может зависеть не только от наступления события, но и от выполнения определенного условия, называемого сторожевым. Объект перейдет из одного состояния в другое в том случае, если произошло указанное событие и сторожевое условие приняло значение "истина".

До срабатывания перехода объект находится в предыдущем от него состоянии, называемым исходным, или в источнике (не путать с начальным состоянием - это разные понятия), а после его срабатывания объект находится в последующем от него состоянии (целевом состоянии).

На диаграмме состояний переход изображается сплошной линией со стрелкой, которая выходит из исходного состояния и направлена в целевое состояние. Каждый переход может быть помечен строкой текста, которая имеет следующий общий формат:<имя события>'('<список параметров>')''['<сторожевое условие>']' <выражение действия>

Событие (event) - спецификация существенных явлений в поведении системы, которые имеют местоположение во времени и пространстве.

Формально, событие представляет собой спецификацию факта, имеющего место в пространстве и во времени. Про события говорят, что они "происходят", при этом отдельные события должны быть упорядочены во времени. После наступления события нельзя уже вернуться к предыдущим, если такая возможность явно не предусмотрена в модели.

Семантика понятия события фиксирует внимание на внешних проявлениях качественных изменений, происходящих при переходе моделируемого объекта из состояния в состояние.

Например, при включении электрического переключателя происходит событие, в результате которого комната освещается. После успешного ремонта компьютера также происходит немаловажное событие - восстановление его работоспособности. Если поднять трубку обычного телефона, то, в случае его исправности, мы ожидаем услышать тоновый сигнал. Это тоже является событием.

В языке UML события играют роль стимулов, которые инициируют переходы из одних состояний в другие. В качестве событий можно рассматривать сигналы, вызовы, окончание фиксированных промежутков времени или моменты окончания выполнения определенных действий. В зависимости от вида происходящих событий - стимулов в языке UML различают два типа переходов: триггерные и нетриггерные.

Переход называется **триггерным**, если его специфицирует событие-триггер, связанное с внешними условиями по отношению к рассматриваемому состоянию.

В этом случае рядом со стрелкой триггерного перехода обязательно указывается имя события в форме строки текста, начинающейся со строчной буквы. Наиболее часто в качестве имен триггерных переходов задают имена операций, вызываемых у тех или иных объектов системы. После имени такого события следуют круглые скобки для явного задания параметров соответствующей операции. Если таких параметров нет, то список параметров со скобками может отсутствовать. Например, переход на рис. 46, а, является триггерным, поскольку с ним связано конкретное событие-триггер, происходящее асинхронно при срабатывании некоторого датчика.

Переход называется **нетриггерным**, если он происходит по завершении выполнения ду-деятельности в данном состоянии.

Нетриггерные переходы часто называют переходами по завершении ду-деятельности. Для них рядом со стрелкой перехода не указывается никакого имени события, а в исходном состоянии должна быть описана внутренняя ду-деятельность, по окончании которой произойдет тот или иной нетриггерный переход.

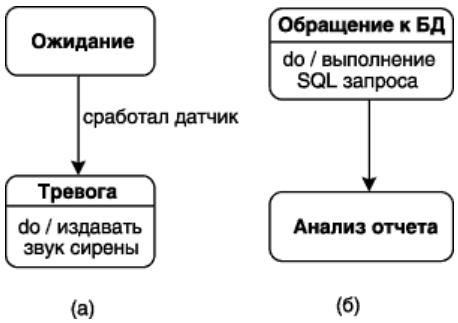


Рис. 46. Графическое изображение триггерного (а) и нетриггерного (б) переходов на диаграмме состояний

Сторожевое условие(guardcondition) - логическое условие, записанное в прямых скобках и представляющее собой булевское выражение.При этом булевское выражение должно принимать одно из двух взаимно исключающих значений: "истина" или "ложь". Из контекста диаграммы состояний должна явно следовать семантика этого выражения, а для записи выражения может использоваться обычный язык, псевдокод или язык программирования.

Дополнение триггерных и нетриггерных переходов сторожевыми условиями позволяет явно специфицировать семантику их срабатывания.

Если сторожевое условие принимает значение "истина", то соответствующий переход при наступлении события-триггера или завершении деятельности может сработать, в результате чего объект перейдет в целевое состояние.

Если же сторожевое условие принимает значение "ложь", то переход не может сработать, даже если произошло событие-триггер или завершилась деятельность в исходном состоянии.

Очевидно, в случае невыполнения сторожевого условия моделируемый объект или система останется в исходном состоянии. Однако вычисление истинности сторожевого условия в модели происходит только после возникновения ассоциированного с ним события-триггера или завершения деятельности, которые инициируют соответствующий переход.

Поскольку общее количество выходящих переходов из любого состояния не ограничено, не исключена

ситуация, когда из одного состояния могут выходить несколько переходов с идентичным событием-триггером. Каждый такой переход должен содержать собственное сторожевое условие, при этом никакие два или более сторожевых условий не должны одновременно принимать значение "истина". В противном случае на диаграмме состояний возникнет конфликт триггерных переходов, что делает несостоятельной (illformed) модель системы в целом.

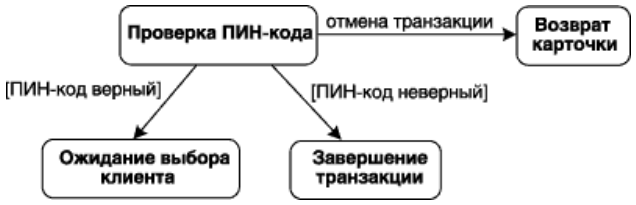


Рис. 47. Триггерные и нетриггерные переходы на диаграмме состояний

Изображенный фрагмент диаграммы состояний (рис. 47) моделирует изменение состояний банкомата при проверке ПИН-кода. Нетриггерные переходы на данной диаграмме помечены сторожевыми условиями, которые исключают конфликт между ними. Что касается триггерного перехода, помеченного событием отмена транзакции, то он происходит независимо от проверки ПИН-кода в том случае, когда клиент решил отказаться от ввода ПИН-кода.

Выражение действия(actionexpression) представляет собой вызов операции или передачу сообщения, имеет атомарный характер и выполняется сразу после срабатывания соответствующего перехода до начала действий в целевом состоянии. Выражение действия выполняется в том и только в том случае, когда переход срабатывает.

Атомарность действия означает, что оно не может быть прервано никаким другим действием до тех пор, пока не закончится его выполнение. Данное действие может оказывать влияние как на сам объект, так и на его окружение, если это с очевидностью следует из контекста модели. Данное выражение записывается после знака "/" в строке текста, присоединенной к соответствующему переходу.

В общем случае, выражение действия может содержать целый список отдельных действий, разделенных символом ";". Обязательное требование - все действия из списка должны четко различаться между собой и следовать в порядке их записи. На синтаксис записи выражений действия не накладывается никаких ограничений. Главное - их запись должна быть понятна разработчикам модели и программистам. Поэтому чаще всего выражения записывают на одном из языков программирования, который предполагается использовать для реализации модели.

В качестве примера выражения действия перехода (рис. 48) может служить отображение сообщения на экране банкомата в том случае, когда запрашиваемая клиентом сумма превосходит остаток на его счету. В случае если кредит не превышен, то происходит переход в состояние получения наличных.

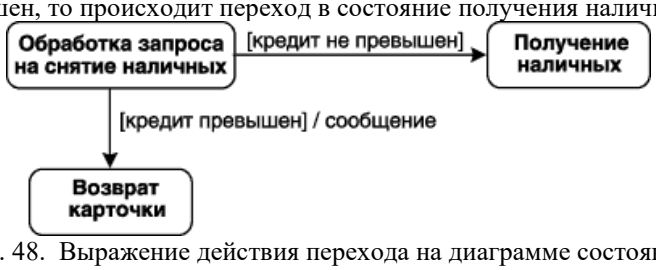


Рис. 48. Выражение действия перехода на диаграмме состояний

Формализм конечных автоматов допускает вложение одних конечных автоматов в другие для уточнения внутренней структуры отдельных более общих состояний. В этом случае вложенные конечные автоматы получили название конечных подавтоматов. Подавтоматы могут использоваться для внутренней спецификации процедур и функций, реализация которых обусловливает поведение моделируемой системы или объекта.

Моделирование сложных объектов и систем связано с многоуровневым представлением их состояний. В этом случае возникает необходимость детализировать отдельные состояния, сделав их составными.

Составное состояние (compositestate) - сложное состояние, которое состоит из других вложенных в него состояний (состояние-композит). Вложенные состояния выступают по отношению к составному состоянию как **подсостояния** (substate). И хотя между ними имеет место отношение композиции, графически все вершины диаграммы, которые соответствуют вложенным состояниям, изображаются внутри символа составного состояния (рис. 49). В этом случае размеры графического символа составного состояния увеличиваются, так чтобы вместить в себя все подсостояния.

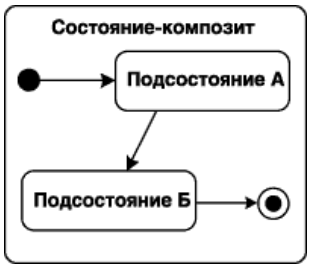


Рис. 49. Графическое представление составного состояния с двумя вложенными в него последовательными подсостояниями

Составное состояние может содержать или несколько последовательных подсостояний, или несколько параллельных конечных подавтоматов. Каждое состояние-композит может уточняться только одним из указанных способов. При этом любое из подсостояний, в свою очередь, может быть состоянием-композитом и содержать внутри себя другие вложенные подсостояния. Количество уровней вложенности составных состояний в языке UML не фиксировано.

Последовательные подсостояния (sequentialsubstates) - вложенные состояния состояния-композита, в рамках которого в каждый момент времени объект может находиться в одном и только одном подсостоянии.

Поведение объекта в этом случае представляет собой последовательную смену подсостояний, от начального до конечного. Моделируемый объект или система продолжает находиться в составном состоянии, тем не менее, введение в рассмотрение последовательных подсостояний позволяет учесть более тонкие логические аспекты его внутреннего поведения.

В качестве примера моделируемой системы стоит рассмотреть обычный телефонный аппарат. Он может находиться в различных состояниях, в частности в состоянии дозвона до абонента. Очевидно, для того чтобы позвонить, необходимо снять телефонную трубку, услышать тоновый сигнал, после чего набрать нужный телефонный номер. Таким образом, состояние дозвона до абонента является составным и состоит из двух последовательных подсостояний: Телефонная трубка поднята и Набор телефонного номера. Фрагмент диаграммы состояний для этого примера содержит одно состояние-композит, которое состоит из двух последовательных подсостояний (рис. 50).

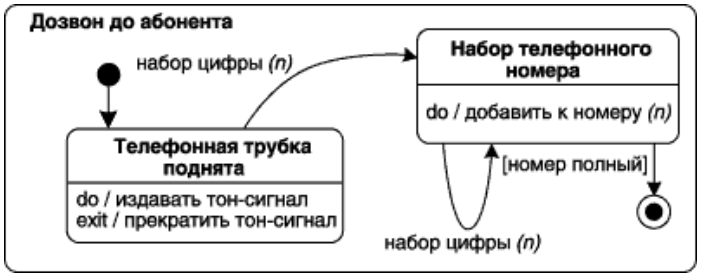


Рис. 50. Пример составного состояния с двумя вложенными последовательными подсостояниями

Некоторых пояснений могут потребовать переходы. Два из них специфицируют событие-триггер, которое имеет имя: набор цифры(n) с параметром n. В качестве параметра, как нетрудно предположить, выступает отдельная цифра на диске телефонного аппарата. Переход из начального подсостояния не содержит никакой строки текста. Последний переход в конечное подсостояние также не имеет события-триггера, но имеет сторожевое условие, проверяющее полноту набранного номера абонента. Только в случае истинности этого условия телефонный аппарат может перейти в конечное состояние для состояния-композита Дозвон до абонента.

Каждое составное состояние должно содержать в качестве вложенных состояний начальное и конечное состояния. При этом начальное подсостояние является исходным, когда происходит переход объекта в данное составное состояние. Если составное состояние содержит внутри себя конечное состояние, то переход в это вложенное конечное состояние означает завершение нахождения объекта в данном составном состоянии. Важно помнить, что для последовательных подсостояний начальное и конечное состояния должны быть единственными в каждом составном состоянии.

Это можно объяснить следующим образом. Каждая совокупность вложенных последовательных подсостояний представляет собой конечный подавтомат того конечного автомата, которому принадлежит рассматриваемое составное состояние. Поскольку каждый конечный автомат может иметь по определению единственное начальное и единственное конечное состояния, то для любого его конечного подавтомата это

условие также должно выполняться.

Параллельные подсостояния (concurrentsubstates) - вложенные состояния, используемые для спецификации двух и более конечных подавтоматов, которые могут выполняться параллельно внутри составного состояния.

Каждый из конечных подавтоматов занимает некоторую графическую область внутри составного состояния, которая отделяется от остальных горизонтальной пунктирной линией. Если на диаграмме состояний имеется составное состояние с вложенными параллельными подсостояниями, то объект может одновременно находиться в каждом из этих подсостояний.

Отдельные параллельные подсостояния могут, в свою очередь, состоять из нескольких последовательных подсостояний (рис. 51). В этом случае по определению моделируемый объект может находиться только в одном из последовательных подсостояний каждого подавтомата. Таким образом, для фрагмента диаграммы состояний (рис. 51) допустимо одновременное нахождение объекта только в следующих подсостояниях: (А, В, Г), (Б, В, Г), (А, В, Д), (Б, В, Д).

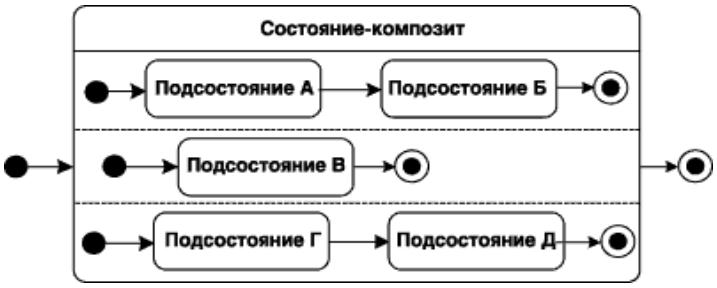


Рис. 51. Графическое изображение состояния-композита с вложенными параллельными подсостояниями

Несовместимое подсостояние (disjointsubstate) - подсостояние, в котором подсистема не может находиться одновременно с другими подсостояниями одного и того же составного состояния.

В этом контексте недопустимо нахождение объекта одновременно в несовместимых подсостояниях (А, Б, В) или (В, Г, Д).

Поскольку каждый регион вложенного состояния специфицирует некоторый конечный подавтомат, то для каждого из вложенных конечных подавтоматов могут быть определены собственные начальное и конечное состояния.

При переходе в данное составное состояние каждый из конечных подавтоматов оказывается в своем начальном состоянии. Далее происходит параллельное выполнение каждого из этих конечных подавтоматов, причем выход из составного состояния будет возможен лишь в том случае, когда все конечные подавтоматы будут находиться в своих конечных состояниях. Если какой-либо из конечных подавтоматов пришел в свое финальное состояние раньше других, то он должен ожидать, пока и другие подавтоматы не придут в свои финальные состояния.

В некоторых случаях бывает желательно скрыть внутреннюю структуру составного состояния. Например, отдельный конечный подавтомат, специфицирующий составное состояние, может быть настолько большим по масштабу, что его визуализация затруднит общее представление диаграммы состояний. В подобной ситуации допускается не раскрывать на исходной диаграмме состояний данное составное состояние, а указать в правом нижнем углу специальный символ-пиктограмму (рис. 52).

В последующем диаграмма состояний для соответствующего конечного подавтомата может быть изображена отдельно от основной диаграммы с необходимыми комментариями.



Рис. 52. Составное состояние со скрытой внутренней структурой и специальной пиктограммой

Обычный конечный автомат не позволяет учитывать предысторию в процессе моделирования поведения систем и объектов. Однако функционирование ряда систем основано на возможности выхода из отдельного состояния-композита с последующим возвращением в это же состояние.

Может оказаться необходимым учесть ту часть деятельности, которая была выполнена на момент выхода из

этого состояния-композита, чтобы не начинать ее выполнение сначала. Для этой цели в языке UML существует историческое состояние.

Историческое состояние (historystate) - псевдосостояние, используемое для запоминания того из последовательных подсостояний, которое было текущим в момент выхода из составного состояния.

Историческое состояние применяется только в контексте составного состояния. При этом существует две разновидности исторического состояния: неглубокое или недавнее и глубокое или давнее (рис. 53).

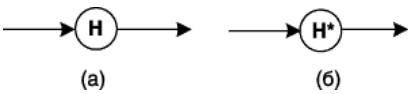


Рис. 53. Графическое изображение недавнего (а) и давнего (б) исторического состояния

Неглубокое историческое состояние (shallowhistorystate) обозначается в форме небольшой окружности, в которую помещена латинская буква "H" (рис. 53, а). Это состояние обладает следующей семантикой. Во-первых, оно является первым подсостоянием в составном состоянии, и переход извне в рассматриваемое составное состояние должен вести непосредственно в данное историческое состояние. Во-вторых, при первом попадании в неглубокое историческое состояние оно не хранит никакой истории. Другими словами, при первом переходе в недавнее историческое состояние оно заменяет собой начальное состояние соответствующего конечного подавтомата.

Далее могут последовательно изменяться вложенные подсостояния. Если в некоторый момент происходит выход из составного состояния (например, в случае наступления некоторого события), то рассматриваемое историческое состояние запоминает то из подсостояний, которое было текущим на момент выхода из данного составного состояния. При последующем входе в это составное состояние неглубокое историческое подсостояние имеет непустую историю и сразу отправляет конечный подавтомат в запомненное подсостояние, минуя все предшествующие ему подсостояния.

Историческое состояние теряет свою историю в тот момент, когда конечный подавтомат доходит до своего конечного состояния. При этом неглубокое историческое состояние запоминает историю только того конечного подавтомата, к которому оно относится. Другими словами, этот тип псевдосостояния способен запомнить историю только одного с ним уровня вложенности.

Глубокое историческое состояние (deephistorystate или состояние глубокой истории) также обозначается в форме небольшой окружности, в которую помещена латинская буква "H" с дополнительным символом "*" (рис. 53, б), и служит для запоминания всех подсостояний любого уровня вложенности для исходного составного состояния.

В отдельных случаях возникает необходимость явно показать ситуацию, когда переход может иметь несколько исходных состояний или целевых состояний. Такой переход получил название - **параллельный переход**. Введение в рассмотрение параллельных переходов может быть обусловлено необходимостью синхронизировать и/или разделить отдельные процессы управления на параллельные нити без спецификации дополнительной синхронизации в параллельных конечных подавтоматах.

Графически такой переход изображается вертикальной черточкой, аналогично обозначению перехода в известном формализме сетей Петри. Если параллельный переход имеет две или более исходящих из него дуг (рис. 54, а), то его называют разделением (fork). Если же он имеет две или более входящие дуги (рис. 54, б), то его называют слиянием (join). Текстовая строка спецификации параллельного перехода записывается рядом с черточкой и относится ко всем входящим или исходящим дугам.

Срабатывание параллельного перехода: в первом случае происходит разделение составного конечного автомата на два конечных подавтомата, образующих параллельные ветви вложенных подпроцессов. При этом после срабатывания перехода-разделения моделируемая система или объект одновременно будет находиться во всех целевых подсостояниях этого параллельного перехода (подсостояния 1 и 2). Далее процесс изменения состояний будет протекать согласно ранее рассмотренным правилам для составных состояний.



Рис. 54. Графическое изображение перехода-разделения в параллельные подсостояния (а) и перехода-слияния из параллельных подсостояний (б)

Во втором случае переход-слияние срабатывает, если имеет место событие-триггер для всех исходных состояний этого перехода, и выполнено (при его наличии) сторожевое условие. При срабатывании перехода-слияния одновременно покидаются все исходные подсостояния перехода (подсостояния 3 и 4) и происходит переход в целевое состояние. При этом каждое из исходных подсостояний перехода должно принадлежать отдельному конечному подавтомату, входящему в составной конечный автомат (процессу Б).

Переход, стрелка которого соединена с границей составного состояния, обозначает переход в это составное состояние (переход a на рис. 55). Он эквивалентен переходу в начальное состояние каждого из конечных подавтоматов (единственному на рис. 55), входящих в состав данного состояния-композита.

Переход f, выходящий из составного состояния (рис. 55), относится к каждому из вложенных состояний. Это означает, что моделируемая система или объект при наступления события f может покинуть данное составное состояние, находясь в любом из его вложенных состояний В и Г.

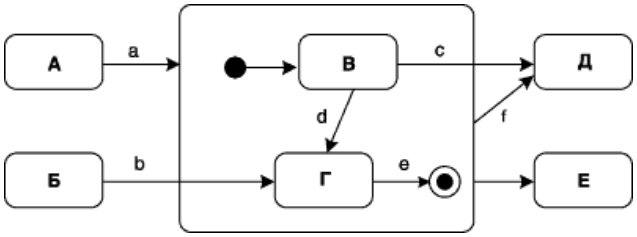


Рис. 55. Варианты переходов в составное состояние и из составного состояния

Иногда желательно реализовать ситуацию, когда выход из отдельного вложенного состояния соответствовал бы также выходу из составного состояния. В этом случае изображают переход, который непосредственно выходит из вложенного состояния и пересекает границу состояния-композита (переход c на рис. 55). Аналогично, допускается изображение переходов, входящих извне состояния-композита в отдельное вложенное состояние (переход b на рис. 55).

Переход d является внутренним для рассматриваемого состояния-композита и никак не влияет на выход из состояния-композита. Выход из данного составного состояния также возможен при наступлении события e, которое приводит в его конечное состояние, а из него - в состояние Е, находящееся вне данного состояния-композита.

Состояние синхронизации (synchstate) - псевдосостояние в конечном автомате, которое используется для синхронизации параллельных областей конечного автомата.

Синхронизирующее состояние обозначается небольшой окружностью, внутри которой помещен символ звездочки "*". Оно используется совместно с переходом-слиянием или переходом-разделением для того, чтобы явно указать события в других конечных подавтоматах, оказывающие непосредственное влияние на поведение данного подавтомата.

Так, например, при включении компьютера с некоторой сетевой операционной системой параллельно начинается выполнение нескольких процессов. В частности, происходит проверка пароля пользователя и запуск различных служб. При этом работа пользователя на компьютере станет возможной только в случае успешной его аутентификации, в противном случае компьютер может быть выключен.

Рассмотренные особенности синхронизации этих параллельных процессов учтены на соответствующей диаграмме состояний с помощью синхронизирующего состояния (рис. 56).

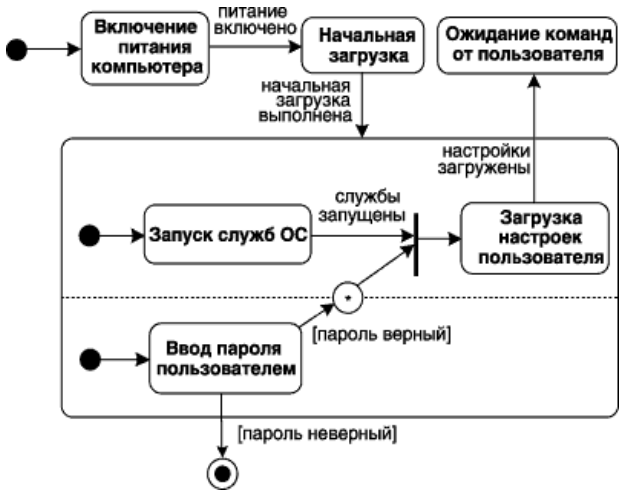


Рис. 56. Диаграмма состояний для примера включения компьютера

Наличие у системы нескольких состояний, отличающихся от простых, таких как "исправен - неисправен", "активен - неактивен", "ожидание - реакция на внешние действия", служит признаком необходимости построения диаграммы состояний. В качестве начального варианта диаграммы состояний, если нет очевидных соображений по поводу состояний объекта, можно воспользоваться подобными состояниями, в качестве составных, уточняя их (детализируя их внутреннюю структуру) по мере рассмотрения логики поведения моделируемой системы или объекта.

При выделении состояний и переходов следует помнить, что длительность срабатывания отдельных переходов должна быть существенно меньшей, чем нахождение моделируемых элементов в соответствующих состояниях. Каждое из состояний должно характеризоваться определенной устойчивостью во времени. Другими словами, из каждого состояния на диаграмме не может быть самопроизвольного перехода в какое бы то ни было другое состояние.

Все переходы должны быть явно специфицированы, в противном случае построенная диаграмма состояний является либо неполной (неадекватной), либо ошибочной с точки зрения нотации языка UML (illformed).

При разработке диаграммы состояний нужно постоянно следить, чтобы объект в каждый момент мог находиться только в единственном состоянии. Если это не так, то данное обстоятельство может быть как следствием ошибки, так и неявным признаком наличия параллельности поведения у моделируемого объекта. В последнем случае следует явно специфицировать необходимое число конечных подавтоматов, вложив их в то составное состояние, которое характеризуется нарушением условия одновременности.

Пример: Программное средство представляет собой базу данных и обеспечивает корректировку данных, а именно в БД «Группы» может изменяться перечень предметов в соответствие с курсом группы и отделением; в БД «Предметы» могут изменяться номера аудиторий и фамилии преподавателей; осуществляет поиск по ФИО преподавателя, номеру аудитории, названию предмета и номеру группы. Программное средство составляет расписание работы для конкретного преподавателя на неделю; для группы на неделю; для группы по конкретному предмету, а также отчет о загрузке аудиторий на каждый день.

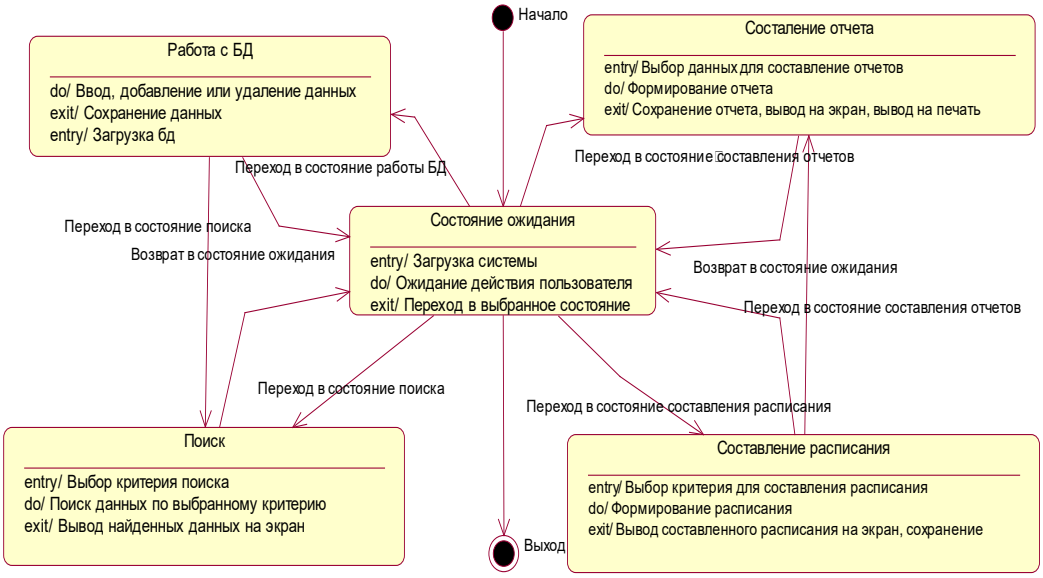


Рис. 57. Пример диаграммы состояний

Порядок выполнения работы

1. Изучить теоретические сведения по теме “Диаграмма состояний”.
2. Разработать диаграмму состояний для произвольной системы индивидуального задания (Приложение А).
3. Оформить отчет, включив в него описание всех компонентов диаграммы состояний согласно индивидуальному варианту задания, ответы на контрольные вопросы.

Контрольные вопросы

1. Дайте определение понятию «диаграмма состояний».
2. Опишите назначение диаграммы состояний.
3. Дайте определение понятию «конечный автомат».
4. Дайте определение понятиям «состояния», «действие», «псевдосостояние». Графическое изображение состояния.
5. Что представляет собой переход? Какие бывают переходы, в чем их различие?
6. Дайте определение понятию «событие».
7. Дайте определения следующим понятиям: «составное состояние», «последовательные подсостояния», «параллельные подсостояния», «несовместимое подсостояние», «историческое состояние», «параллельный переход», «состояние синхронизации».

Критерии оценки:
Всего баллов – 5 (Максимально)
Выполнение заданий лабораторной работы может быть оценено от 0 до 3 баллов.
Ответы на контрольные вопросы – от 0 до 2 баллов.

Задания для самостоятельной работы студентов:

Для каждого из предложенных ниже вариантов требуется при помощи CASE-средств построить модель программного обеспечения с использованием объектного подхода. Модель программного обеспечения включает в себя следующие канонические диаграммы:

- вариантов использования (usecasediagram);
- классов (classdiagram);
- кооперации (collaborationdiagram);
- последовательности (sequencediagram);
- состояний (statechartdiagram);
- деятельности (activitydiagram);

- компонентов (componentdiagram);
- развертывания (deploymentdiagram).

Все канонические диаграммы модели программного обеспечения должны удовлетворять перечисленным ниже требованиям:

- каждая диаграмма должна служить законченным представлением соответствующего фрагмента моделируемой предметной области;
- все сущности на диаграмме модели должны быть одного уровня представления;
- вся информация о сущностях должна быть явно представлена на диаграммах;
- диаграммы не должны содержать противоречивой информации;
- каждая диаграмма должна быть самодостаточной для правильной интерпретации всех ее элементов и понимания семантики всех используемых графических символов;
- любая модель системы должна содержать только те элементы, которые определены в нотации языка UML.

На диаграммах модели программного обеспечения все соответствующие компоненты должны сопровождаться описанием. Эти описания должны быть емкими и составляться на русском языке. Описание некоторых компонентов диаграмм должно включать в себя пояснение, предусловие, потоки событий (если таковые есть) и постусловие. Помимо описания компонентов на канонических диаграммах должны быть явно указаны стереотипы связей и сообщений между различными компонентами.

Все канонические диаграммы модели программного обеспечения должны быть оформлены в виде одного общего отчета в той последовательности, в которой они приведены в лабораторном практикуме, либо в виде отдельных отчетов по каждой из лабораторных работ.

Если диаграммы оформлены в виде общего отчета, то он должен включать в себя:

- условие индивидуального задания в соответствие с вариантом, выданным преподавателем;
- глоссарий проекта, оформленный в виде таблицы;
- пояснения каждой канонической диаграммы модели и ее графическое представление (скриншот) исходя из условия индивидуального задания;
- выводы по готовому проекту.

Отчет оформляется в Microsoft Word или в любом другом текстовом редакторе и предоставляется преподавателю в печатном виде после окончания выполнения данной лабораторной работы для отдельных отчетов, либо по окончанию всех лабораторных работ для общего отчета для его защиты.

Вариант 1. Цифровой диктофон

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения цифрового диктофона.

Требуется разработать средствами UMLмодель программного обеспечения, управляющего работой цифрового диктофона. Цифровой диктофон – это бытовое электронное устройство для записи и воспроизведения речи. Звуковые сообщения записываются через встроенный микрофон и сохраняются в памяти устройства. Сообщения воспроизводятся через встроенный громкоговоритель. Работа устройства осуществляется под управлением центрального процессора.

Диктофон хранит до 10 звуковых сообщений. Длина каждого сообщения ограничена размером свободной памяти. Диктофон осуществляет прямой (по номеру сообщения) доступ к любому сообщению из памяти. Пользователь имеет возможность воспроизводить сообщения, хранящиеся в памяти диктофона, стирать их, записывать новые. Исполнителем должна быть разработана схема базы данных для хранения сообщений в памяти диктофона.

Интерфейс с пользователем осуществляется при помощи экранного меню и управляющих кнопок на корпусе диктофона. При помощи кнопок-стрелок осуществляется навигация по пунктам меню. Кнопки «Да», «Нет» служат для подтверждения или отмены пользователем выбора той или иной опции меню (структуру меню исполнитель должен разработать самостоятельно). Имеются также кнопки «Воспроизведение», «Пауза» и «Запись» для работы со звуковыми сообщениями. Во время записи сообщения на экране отображается время, в течение которого ведется запись, при воспроизведении – длительность воспроизведенной части сообщения.

Если диктофон не используется, через 30 секунд он автоматически переходит в режим сбережения энергии. В этом режиме никакие операции над звуковыми сообщениями не возможны. Энергия расходуется только на сохранение памяти диктофона в неизменном состоянии. Переход из режима сбережения энергии в обычный

режим осуществляется при нажатии пользователем любой кнопки.

В диктофоне имеется датчик уровня заряда батарей. При падении уровня заряда ниже установленного предела диктофон автоматически переходит в режим сбережения энергии (независимо от того используется он в данный момент или нет). Переход в обычный режим становится возможным только после восстановления нормального уровня заряда батарей.

Вариант 2. Торговый автомат

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения торгового автомата.

Требуется разработать средствами UMLмодель программного обеспечения встроенного процессора универсального торгового автомата.

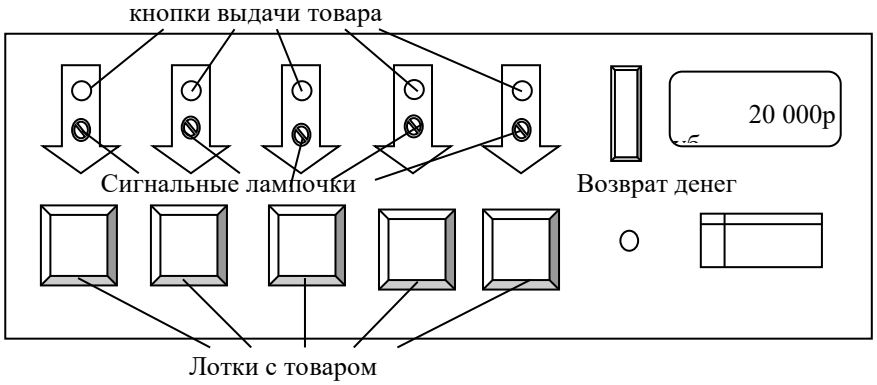


Рис. 85. Лицевая панель торгового автомата

Внешний вид автомата изображен на рисунке. В автомате имеется пять лотков для хранения и выдачи товаров. Загрузка товаров на лотки осуществляется обслуживающим персоналом. Автомат следит за наличием товара. Если какой-либо товар распродан, автомат отправляет сообщение об этом на станцию обслуживания и информирует покупателей (зажигается красная лампочка рядом с лотком данного товара).

Автомат принимает к оплате бумажные купюры и монеты. Специальный индикатор высвечивает текущую сумму денег, принятых автоматом к оплате. После ввода денег клиент нажимает на кнопку выдачи товара. Выдача товара производится только в том случае, если введенная сумма денег соответствует цене товара. Товар выдается поштучно. При нажатии на кнопку «Возврат» клиенту возвращаются все принятые от него к оплате деньги. Возврат денег не производился после выдачи товара. Автомат должен корректно работать при одновременном нажатии на кнопки выдачи товара и возврата денег. В специальном отделении автомата, закрываемом замком, есть «секретная кнопка», которая используется обслуживающим персоналом для выемки выручки. При нажатии на эту кнопку открывается доступ к ящику с деньгами. Автомат получает со станции обслуживания данные о товарах и хранит их в своей памяти. Данные включают в себя цену, наименование товара, номер лотка, на котором находится товар и количество товара на лотке.

Необходимо разработать модель программного средства об учете и хранении товаров.

Вариант 3. Табло на станции метро

Тема: Объектно-ориентированный анализ н проектирование программного обеспечении табло на станции метро.

Требуется разработать средствами UMLмодель программного обеспечения табло для информационной службы метрополитена.

Табло расположены на каждой станции метро. Они работают под управлением единого пункта управления (ПУ) информационной службы метро. Табло отображает текущее время (часы, минуты, секунды) и время, прошедшее с момента отправления последнего поезда (минуты, секунды). Момент прибытия и отправления поезда определяется при помощи датчиков, устанавливаемых не путях. Все табло метро синхронизованы, текущее время отсчитывается и устанавливается из центральной службы времени, находящейся на ПУ.

На табло высвечивается конечная станция назначения прибывающего поезда. Эти данные содержатся в расписании движения поездов, которое хранится и памяти табло и периодически обновляется с ПУ.

В «бегущей строке» табло отображается рекламная информация. Память табло хранит до 10 рекламных сообщений. Сообщения отображаются друг за другом с небольшими паузами, циклически. Содержание

рекламных сообщений поступает с ПУ.

Дополнительная функция табло - по запросу с ПУ оно пересылает данные о нарушениях расписания (преждевременных отправлениях поездов или опозданиях).

Пояснение**:** в задании требуемся разработать модель ПО только для табло, но не для пункта управления информационной службы.

В ходе выполнения задания необходимо разработать модель программного средства для хранения рекламных сообщений, расписания и сведений о нарушении расписания.

Вариант 4. Система автоматизации дли пункта проката видеокассет

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения системы автоматизации для пункта проката видеокассет.

Требуется разработать средствами UMLмодель программной системы автоматизации работы пункта проката видеокассет (далее в тексте системы).

Пункт проката содержит каталог кассет, имеющихся в наличии в данный момент времени. Система поддерживает работу каталога, позволяя служащим проката добавлять новые наименования кассет, удалять старые и редактировать данные о кассетах.

Клиент, обратившийся в пункт, выбирает кассету по каталогу, вносит залог и забирает ее на определенный срок. Срок проката, измеряемый в сутках, оговаривается при выдаче кассеты. Стоимость проката вычисляется системой исходя из тарифа за сутки и срока проката. Клиент возвращает кассету и оплачивает прокат. Если кассета не повреждена, клиенту возвращается залог. Служащий пункта проката регистрирует сдачу кассеты клиенту и ее возврат в системе. Если клиент повредил кассету, то кассета удаляется из каталога, а залог остается в кассе проката. При необходимости служащий может запросить у системы следующие данные:

- имеется ли в наличии кассета с данным названием;
- когда будет возвращена какая-либо кассета из тех, что сданы в прокат;
- является ли данный клиент постоянным клиентом пункта проката (пользовался ли прокатом 5 или более раз).

Постоянным клиентам предоставляются скидки, а также от них принимаются заявки на пополнение ассортимента кассет. Заявки регистрируются в системе, по ним готовится итоговый отчет, руководствуясь которым, служащие пункта проката обновляют ассортимент кассет.

Необходимо разработать модель программного средства для хранения каталога, учетных записей о прокате кассет и заявок на пополнение ассортимента.

Вариант 5. Мини-АТС

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения мини-АТС.

Требуется разработать средствами UMLмодель программного обеспечения встроенного микропроцессора учрежденческой мини-АТС (автоматической телефонной станции).

Мини-АТС осуществляет связь между служащими учреждения. Каждый абонент подключен к ней линией связи. Мини-АТС соединяет линии абонентов (осуществляет коммутацию линий). Абоненты имеют номера, состоящие из трех цифр. Специальный номер «9» зарезервирован для внешней связи.

Телефонное соединение абонентов производится следующим образом. Абонент поднимает трубку телефона, и мини-АТС получает сигнал «Трубка». В ответ мини-АТС посылает сигнал «Тон». Приняв этот сигнал, абонент набирает телефонный номер (посылает три сигнала «Цифра»). Мини-АТС проверяет готовность вызываемого абонента. Если абонент не готов (его линия занята), мини-АТС посылает вызывающему абоненту сигнал «Занято». Если абонент готов, мини-АТС посылает обоим абонентам сигнал «Вызов». При этом телефон вызываемого абонента начинает звонить, а вызывающий абонент слышит в трубке длинные гудки. Вызываемый абонент снимает трубку, и мини-АТС получает от него сигнал «Трубка», после чего осуществляет коммутацию линии. Абоненты обмениваются сигналами «Данные», которые мини-АТС должна передавать от одного абонента к другому. Когда один из абонентов опускает трубку, мини-АТС получает сигнал «Конец» и посылает другому абоненту сигнал «Тон».

В любой момент абонент может положить трубку, при этом мини-АТС получает сигнал «Конец». После получения этого сигнала сеанс обслуживания абонента завершается.

Если абонент желает соединиться с абонентом за пределами учреждения, то он набирает номер «9». Мини-АТС посылает по линии, соединяющей с внешней (городской) АТС, сигнал «Трубка» и в дальнейшем служит

посредником между телефоном абонента и внешней АТС. Она принимает и передает сигналы и данные между ними, не внося никаких изменений. Единственное исключение касается завершения сеанса. Получив от городской АТС сигнал «Конец», мини-АТС посылает абоненту сигнал «Тон» и ждет сигнала «Конец» для завершения обслуживания абонента. Если вызывавший абонент первым вешает трубку, то мини-АТС получает сигнал «Конец» и передает его городской АТС и завершает сеанс. Мини-АТС может получить сигнал «Вызов» от городской АТС. Это происходит, когда нет соединений с внешними абонентами. Сигнал «Вызов» от городской АТС передается абоненту с кодом «000». Только этот абонент может отвечать на внешние звонки.

Вариант 6. Телефон

Тема: Объектно-ориентированный анализ и проектированиепрограммного обеспечение встроенного микропроцессора телефона.

Требуется разработать средствами UMLмодель программного обеспечения встроенного микропроцессора для аппарата учрежденческой телефонной сети.

Аппарат подключен к линии связи, ведущей к мини-АТС. В его задачу входит прием и передача сигналов (в том числе и голосовых данных) мини-АТС. Аппарат имеет кнопочную панель управления, экран для отображения набираемых номеров, звонок и трубку, в которую встроены микрофон и громкоговоритель.

В начальном состоянии трубка телефона повешена, телефон не реагирует на нажатия кнопок. Телефон реагирует только на сигнал «Вызов» от мини-АТС, при этом включается звонок.

При снятии трубки на АТС подается сигнал «Трубка». При получении ответного сигнала «Тон» от АТС телефон воспроизводит звуковой тон «Готов» (длинный непрекращающийся гудок) в трубку. При получении сигнала «Занято», н трубке воспроизводится тон «Занято» (частые короткие гудки).

Пользователь, слыша в трубке тон «Готов», набирает трехзначный номер. Номер может быть набран при помощи кнопок с цифрами или нажатием на специальную кнопку «#». При нажатии на кнопку с цифрой соответствующий ей сигнал «Цифра» передается АТС. Нажатия на кнопки с цифрами после третьей кнопки игнорируются. Во время набора номера введенные цифры отображаются на экране. Последний полностью набранный номер запоминается в памяти аппарата для того, чтобы можно было его воспроизвести при нажатии на кнопку «#». При нажатии на эту кнопку номер из памяти аппарата высвечивается на экране, и АТС передается последовательность до трех сигналов «Цифра». В ответ на набранный номер от АТС приходит либо сигнал «Занято», либо сигнал «Вызов». При получении сигнала «Вызов» телефон воспроизводит в трубку длинные гудки до того момента, когда АТС осуществит коммутацию и передаст сигнал «Данные». Телефон воспроизводит данные, передаваемые с сигналом, в трубку. Ответ пользователя воспринимается микрофоном трубки, преобразуется в сигнал «Данные» и передается АТС. Обмен данными прерывается, если повешена трубка одного из телефонов, участвующих в обмене. О том, что трубку повесил вызываемый абонент, сообщает сигнал «Занято», посылаемый АТС. После того, как трубка аппарата была повешена, телефон посылает АТС сигнал «Конец», и телефон переходит в начальное состояние.

Вариант 7. Стиральная машина

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения встроенного микропроцессора стиральной машины.

Требуется разработать средствами UMLмодель программного обеспечения встроенного микропроцессора стиральной машины.

Машина предназначена для автоматической стирки белья. Машина включает в себя следующие устройства: бак для белья, клапаны для забора и слива воды, мотор, устройство подогрева воды, термометр, таймер, дверца для доступа в бак, несколько емкостей для различных моющих средств, панель управления с кнопками и индикатором. В памяти машины хранятся 5 программ стирки, заданные изготовителем. Пользователи не могут вносить в них изменения. Каждая программа определяет температуру воды, длительность стирки, используемые моющие средства (номер емкости и время подачи), скорость вращения бака во время стирки и отжима. Вариант задания предусматривает разработку модели программного обеспечения для хранения программ стирки в памяти машины.

Для использования машины необходимо открыть дверцу, поместить белье в бак, поместить моющие средства в емкости, закрыть дверцу, выбрать программу стирки и нажать на кнопку «Пуск». Перед тем как приступить к стирке машина открывает клапан для забора воды, набирает необходимое количество воды, после чего закрывает клапан. Далее машина действует по выбранной пользователем программе:

- подогревает, если необходимо воду до нужной температуры»;
- включает таймер и запускает вращение бака для стирки;

- по таймеру подает в бак моющие средства, предусмотренные программой;
- по окончании стирки сливает воду и запускает отжим».

Во время работы на индикаторе высвечивается время, прошедшее от начала стирки (минуты и секунды), текущий режим работы (стирка или отжим), номер текущей программы стирки. В целях безопасности дверца бака блокируется до окончания стирки. Машина не воспринимает нажатий на кнопки, за исключением одной – пользователь имеет возможность в любой момент нажать на кнопку «Останов», чтобы принудительно остановить стирку и слить воду.

Вариант 8. Таксофон

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения таксофона.

Требуется разработать средствами UMLмодель встроенной системы управления работой таксофона городской телефонной сети.

Таксофон предназначен для оказания платных услуг телефонной связи. Он подключен к линии связи. В нем имеется кнопочная панель, дисплей, трубка со встроенным микрофоном и громкоговорителем, приемник карт – устройство для считывания телефонных карт, используемых для оплаты разговора.

В начальном состоянии трубка таксофона повешена, дисплей потушен, таксофон не реагирует на нажатия кнопок и какие-либо сигналы из линии. При снятии трубки таксофон выдает на дисплей сообщение «Вставьте карту» и ожидает, когда пользователь вставит карту в приемник. Дальнейшее функционирование таксофона осуществляется только при вставленной карте. Если карту вынимают, таксофон возвращается к началу и выдает сообщение о необходимости вставить карту. При попадании карты в приемник производится считывание информации с карты. Если кредит исчерпан или карта не пригодна (не удается узнать кредит), то таксофон выдает соответствующее сообщение на дисплей таксофона. Если карта может быть использована для оплаты, то на дисплей выдается количество «тарифных единиц» на карте, и на телефонную станцию (АТС) подается сигнал «Трубка». При получении ответного сигнала «Тон» из линии таксофон воспроизводит звуковой гон «Готов» (длинный непрекращающийся гудок) в трубку. При получении сигнала «Занято», в трубке воспроизводится тон «Занято» (короткие гудки).

После получения от АТС сигнала «Тон» от пользователя принимаются семизначный номер вызываемого абонента, остальные нажатия на кнопки игнорируются. Когда пользователь нажимает на кнопку с цифрой соответствующий ей сигнал «Цифра» передается АТС. Во время набора номера введенные цифры отображаются на дисплее. В ответ на набранный номер от АТС приходит либо сигнал «Занято», либо сигнал «Вызов». При получении сигнала «Вызов» таксофон воспроизводит в трубку длинные гудки до того момента, когда АТС осуществит коммутацию и передаст сигнал «Данные». Таксофон воспроизводит данные, передаваемые с сигналом, в трубку. При получении данных из трубки, аппарат преобразует их в сигнал «Данные» и передает их АТС. Во время разговора на дисплее ведется отсчет времени и уменьшается кредит на телефонной карте – каждые 15 секунд вычитается четверть «единицы». Обмен данными прерывается, в следующих случаях:

- исчерпан кредит;
- карта вынута из приемника;
- от АТС пришел сигнал «Занято»;
- повешена трубка таксофона.

Если трубка была повешена, аппарат посылает в линию сигнал «Конец» и выдает на дисплей сообщение «Выньте карту». После извлечения карты из приемника таксофон переходит в начальное состояние.

Вариант 9. Холодильник

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения встроенного процессора холодильника.

Требуется разработать средствами UMLмодель программного обеспечения встроенного процессора холодильника.

Холодильник состоит из нескольких холодильных камер для хранения продуктов. В каждой холодильной камере имеется регулятор температуры, мотор, термометр, индикатор, таймер, датчик открытия двери камеры и устройство для подачи звуковых сигналов.

При помощи терморегулятора устанавливается максимально допустимая температура в данной камере. Мотор предназначен для поддержания низкой температуры. Термометр постоянно измеряет температуру внутри камеры, а индикатор температуры, расположенный на дверце, постоянно высвечивает ее значение. При повышении температуры выше предела, определяемого текущим положением регулятора, включается мотор. При снижении температуры ниже некоторого другого значения, связанного с первым, мотор отключается.

Доступ в камеру осуществляется через дверцу. Если дверь холодильной камеры открыта в течение слишком долгого времени, подается звуковой сигнал. Звуковой сигнал также подается в любых нештатных ситуациях (например, при поломке мотора). Холодильник ведет электронный журнал, в котором отмечаются все происходящие события:

- изменение положения терморегулятора камеры;
- включение и отключение мотора;
- доступ в камеру;
- внештатные ситуации.

Вариантом задания предусмотрена разработка модели программного обеспечения для хранения журнала событий холодильника. Содержимое журнала может быть передано в компьютер, подсоединенный к специальному гнезду на корпусе холодильника.

Вариант 10. Замок

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения встроенного процессора кодового замка.

Требуется разработать средствами UMLмодель программного обеспечения встроенного микропроцессора для кодового замка, регулирующего доступ в помещение.

Кодовый замок состоит из панели с кнопками (цифры «0» … «9», кнопка «Вызов», кнопка «Контроль»), цифрового дисплея, электромеханического замка, звонка. Панель с кнопками устанавливается с наружной стороны двери, замок устанавливается с внутренней стороны двери, звонок устанавливается внутри охраняемого помещения.

В обычном состоянии замок закрыт. Доступ в помещение осуществляется после набора кода доступа, состоящего из четырех цифр. Во время набора кода введенные цифры отображаются на дисплее. Если код набран правильно, то замок открывается на некоторое время, после чего дверь снова закрывается. Содержимое дисплея очищается.

Кнопка «Вызов» используется для подачи звукового сигнала внутри помещения. Кнопка «Контроль» используется для смены кодов. Смена кода доступа осуществляется следующим образом: при открытой двери нужно набрать код контроля, состоящий из четырех цифр, и новый код доступа. Для смены кода контроля нужно при открытой двери и нажатой кнопке «Вызов» набрать код контроля, после чего – новый код контроля.

Вариант 11. Турникет метро

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения встроенного процессора турникета для метрополитена.

Требуется разработать средствами UMLмодель **программного** обеспечения встроенного процессора турникета для метрополитена.

При помощи турникета контролируется проход пассажиров в метро и взимается входная плата. Турникет имеет приемник карт, устройство для перекрывания доступа, таймер, три оптических датчика для определения прохода пассажира, устройство подачи звуковых сигналов, индикаторы «Проход» и **«Стоп».**

В начальном состоянии турникета зажжен индикатор «Стоп», индикатор «Проход» потушен. Если один из датчиков посылает сигнал, то проход через турникет сразу же перекрывается, и подается предупредительный звуковой сигнал. Для прохода пассажир должен поместить карту в приемник карт. **Турникет** считывает с нее данные: срок годности карты и количество «единиц» на ней. Если данные не удается **считать,** либо карта просрочена или заблокирована, то карта возвращается пассажиру, и турникет остается в исходном состоянии. В другом случае с карты списывается одна «единица», карта возвращается из приемника, индикатор «Стоп» гаснет, зажигается индикатор «Проход», и пассажир может пройти через турникет. Получив от одного из датчиков сигнал, турникет ожидает время, отведенное на проход пассажира (5 секунд), после чего он возвращается в начальное состояние.

Наличие трех датчиков в турникете гарантирует, что при проходе пассажира хотя бы один из них подаст сигнал (датчики невозможно перешагнуть, перепрыгнуть и т. д.). Во время прохода пассажира возможна ситуация, когда все три датчика посылают сигналы. В этом случае принимается только первый сигнал и от момента его приема отсчитывается положенное время. Остальные сигналы игнорируются.

Турникет заносит в свою память время всех оплаченных проходов. В конце рабочего дня он передает всю информацию, накопленную за день, в АСУ метрополитена.

В ходе выполнения этого варианта задания должна быть разработана модель программного обеспечения о проходах через турникет.

Вариант 12.Система учета товаров

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения учета товаров.

Требуется разработать средствами UMLмодель системы поддержки заказа и учета товаров в бакалейной лавке.

В бакалейной лавке для каждого товара фиксируется место хранения (определенная полка), количество товара и его поставщик. Система поддержки заказа и учета товаров должна обеспечивать добавление информации о новом товаре, изменение или удаление информации об имеющемся товаре, хранение и редактирование (добавление, изменение и удаление) информации о поставщиках, включающей в себя название фирмы, ее адрес и телефон. При помощи системы составляются заказы для поставщиков. Каждый заказ может содержать несколько позиций, в каждой позиции указываются наименование товара и его количество в заказе. Система учета по требованию пользователя формирует и выдает на печать следующую справочную информацию:

- список всех товаров;
- список товаров, имеющихся в наличии;
- список товаров, количество которых необходимо пополнить;
- список товаров, поставляемых данным поставщиком.

В ходе выполнения этого варианта задания должна быть разработана модель программного обеспечения, хранящего информацию о товарах, заказах и поставщиках.

Вариант 13. Библиотека

Тема: Объектно-ориентированный анализ и проектирование системы автоматизации для библиотеки.

Требуется разработать средствами UMLмодель системы автоматизирующей деятельность библиотеки.

Система поддержки управления библиотекой должна обеспечивать операции (добавление, удаление и изменение) над данными о читателях. В регистрационном списке читателей хранятся следующие сведения: фамилия, имя и отчество читателя; номер его читательского билета и дата выдачи билета. Наряду с регистрационным списком системой должен поддерживаться каталог библиотеки, где хранится информация о книгах: название, список авторов, библиотечный шифр, год и место издания, название издательства, общее количество экземпляров книги в библиотеке и количество экземпляров, доступных в текущий момент времени. Система обеспечивает добавление, удаление и изменение данных каталога, а также поиск книг в каталоге на основании введенного шифра или названия книги. В системе осуществляется регистрация взятых и возвращенных читателем книг. В системе для каждой книге хранится запись о том, кому и когда она была выдана, и когда будет возвращена данная книга. При возврате книги в записи делается соответствующая пометка, а сама запись не удаляется из системы. Система должна выдавать следующую справочную информацию:

- какие книги были выданы за данный промежуток времени;
- какие книги были возвращены за данный промежуток времени;
- какие книги находятся у данного читателя;
- имеется ли в наличии некоторая книга.

Вариант задания предусматривает разработку модели программного средства, хранящего список читателей, каталог книг и записи о выдаче книг.

Вариант14.Интернет-магазин

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения интернет-магазина.

Требуется разработать средствами UMLмодель программного обеспечения Интернет-магазина.

Интернет-магазин позволяет делать покупки с доставкой на дом. Клиенты магазина при помощи программы-браузера имею доступ к каталогу продаваемых товаров, поддержку которого осуществляет администратор Интернет-магазина. В каталоге товары распределены по разделам. О каждом товаре доступна полная информация (наименование товара, вес, цена, изображение, дата изготовления и срок годности). Для удобства клиентов предусмотрена система поиска товаров в каталоге. Заполнение каталога информацией происходит автоматически в начале рабочего дня, информация берется из системы автоматизации торговли.

При выборе клиентами товаров поддерживается виртуальная «торговая корзина». Любое наименование товара может быть добавлено в «корзину» или изъято в любой момент по желанию покупателя с последующим пересчетом общей стоимости покупки. Текущее содержимое «корзины» постоянно показывается клиенту.

По окончании выбора товаров производится оформление заказа и регистрация покупателя. Клиент указывает в регистрационной форме свою фамилию, имя и отчество, адрес доставки заказа и телефон, по которому с ним можно связаться для подтверждения сделанного заказа. Заказы передаются для обработки в систему автоматизации торговли. Проверка наличия товаров на складе и их резервирование Интернет-магазином не производятся.

Вариант 15. WWW-конференция

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения WWW-

конференции.

Требуется разработать средствами UMLмодель программного обеспечения WWW-конференции (форума).

WWW-конференция представляет собой хранилище сообщений в сети Интернет, доступ к которому осуществляется при помощи браузера. Для каждого сообщения конференции хранятся значения следующих полей: номер сообщения, автор, тема, текст сообщения, дата добавления сообщения, ссылка на родительское сообщение. Начальной страницей конференции является иерархический список сообщений. Верхний уровень иерархии составляют сообщения, открывающие новые темы, а подуровни составляют сообщения, полученные в ответ на сообщения верхнего уровня. Сообщение-ответ всегда имеет ссылку на исходное сообщение. В списке отображаются только темы сообщений, их авторы и даты добавления. Просматривая список, пользователь выбирает сообщение и по гиперссылке открывает страницу с текстом сообщения. Помимо текста на этой странице отображается иерархический список сообщений, являющихся ответами, ответами на ответы и т.д. Для удобства пользователей необходимо предусмотреть поиск сообщений по автору или по ключевым словам в теме или тексте сообщения.

Сообщения добавляются в конференцию зарегистрированными пользователями, которые при отправке сообщения должны указать своё имя и пароль. Регистрирует новых пользователей модератор конференции – её ведущий. При регистрации пользователь заполняет специальную форму, содержимое которой затем пересылается модератору и запоминается в базе пользователей. Модератор решает, регистрировать пользователя или нет, и отправляет свой ответ. При добавлении сообщений пользователь имеет возможность начать новую тему или ответить на ранее добавленные сообщения. После добавления сообщения оно доступно для чтения всем пользователям (даже незарегистрированным), и список сообщений обновляется.

Модератор имеет право по тем или иным причинам удалять сообщения любых авторов. Он также может наказывать пользователей, нарушающих правила поведения в конференции, лишая на некоторое время пользователя возможности добавлять и редактировать сообщения.

Вариант задания включает в себя разработку модели программного обеспечения для хранения сообщений конференции и информации об её участниках.

Вариант 16. Каталог ресурсов Интернет

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения каталога ресурсов Интернет.

Требуется разработать средствами UML модель программного обеспечения каталога ресурсов сети Интернет.

В каталоге хранится следующая информация о ресурсах: название ресурса, уникальный локатор ресурса (URL), раздел каталога, в котором содержится ресурс, список ключевых слов, краткое описание, дата последнего обновления, контактная информация.

Доступ пользователей к каталогу осуществляется при помощи браузера. Пользователи каталога могут добавлять новые ресурсы, информация о которых не была внесена ранее. Ресурсы в каталоге классифицируются по разделам. Полный список ресурсов каждого раздела должен быть доступен пользователям. Пользователям каталога должны быть предоставлены возможности по поиску ресурсов. Поиск осуществляется по ключевым словам. Если пользователь не доволен результатами поиска, он может уточнить запрос (осуществить поиск среди результатов предыдущего поиска). Должна быть предусмотрена возможность выдачи результатов поиска в разной форме (вывод всей информации о ресурсах или частичной). Пользователь может отсортировать список ресурсов по релевантности (соответствию ключевым словам из запроса) или по дате обновления. Поскольку содержание ресурсов Интернет со временем изменяется, необходимо следить за датой последнего обновления, периодически опрашивая Web-сайты, URL которых хранятся в каталоге.

Вариант задания включает в себя разработку модели программного средства для хранения и манипулирования информацией о ресурсах.

Вариант 17. Будильник

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения встроенного микропроцессора будильника.

Требуется разработать средствами UMLмодель программного обеспечения встроенного микропроцессора для будильника.

Будильник постоянно отображает текущее время (часы, минуты, например: 12:00). Управление будильником осуществляется следующими кнопками:

- кнопкой режима установки времени»;
- кнопкой режима установки времени срабатывания;
- двумя отдельными кнопками для установки часов и минут;
- кнопкой сброса сиг нала «СБРОС».

На будильнике имеется переключатель режима работы со следующими положениями: «ВЫКЛ», «ВКЛ»,«РАДИО» и «ТАЙМЕР».

Для установки текущего времени нужно нажать на кнопку режима установки и, при нажатой кнопке, нажимать на кнопки установки часов и минут. При каждом нажатии на кнопки, устанавливаемое значение увеличивается на одну единицу (один час или одну минуту соответственно). При достижении максимального значения производится сброс. Для установки времени срабатывания будильника нужно нажать на кнопку режима установки времени срабатывания и, держа кнопку нажатой, нажимать на кнопки установки часов и минут. Когда переключатель режима работы находится в положении «ВКЛ», при достижении времени срабатывания происходит подача звукового сигнала в течение одной минуты. Сигнал можно прервать, нажав на кнопку «СБРОС». При этом сигнал должен быть возобновлен через пять минут. При установке переключателя в положение «ВЫКЛ» звуковой сигнал не подается.

Когда переключатель находится в положении «РАДИО» работает радиоприемник. При переводе переключателя в положение «ТАЙМЕР» включается радиоприемник на тридцать минут, а затем часы переходят в состояние будильника (аналогично положению «ВКЛ»). При нажатии на кнопку режима установки времени, будильник должен отображать время срабатывания.

Вариант 18. Генеалогическое дерево

Тема: Объектно-ориентированный анализ и проектирование программной системы поддержки генеалогических деревьев.

Требуется разработать средствами UMLмодель системы для поддержки генеалогических деревьев.

Система хранит сведения о персонах (Ф.И.О., пол, дата рождения, дата смерти, биография) и о родственных связях между ними. Связи бывают только трех видов: «мужья-жены», «дети-родители» и «братья-сестры». Система обеспечивает возможность добавления данных о новых персонах и родственных связях, изменение введенных данных и удаление ненужных данных. Система следит за непротиворечивостью вводимых данных. Недопустимо, чтобы человек был собственным предком или потомком.

Разработанная модель должна содержать схему базы данных для хранения генеалогических деревьев. Пользователи системы могут осуществлять поиск полезной информации по дереву:

- находить для указанного члена семьи его детей;
- находить для указанного члена семьи его родителей;
- находить для указанной персоны братьев и сестер, если таковые есть;
- получать список всех предков персоны;
- получать список всех потомков персоны;
- получать список всех родственников персоны;
- прослеживать цепочку родственных связей от одной персоны до другой (например, если Петр является шурином Ивана, то на запрос о родственных связях между Петром и Иваном выдается такой результат: «Петр - брат Ольги, Ольга - жена Ивана»).

Вариант 19. Телевизор

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения для управления работой телевизора.

Требуется разработать средствами UMLмодель встроенной системы управления работой телевизора.

В телевизоре имеются следующие устройства: приемник телевизионного сигнала, устройство отображения картинки, память каналов, память настроек, управляющие кнопки, пульт дистанционного управления (ДУ). Управление телевизором осуществляется при помощи кнопок на корпусе (их четыре: «ВКЛ/ВЫКЛ»,«-», «+», кнопка начальной установки) и пульта ДУ. Кнопка «ВКЛ/ВЫКЛ» позволяет включать и выключать телевизор. После включения телевизора на экран отображается передача, идущая по каналу №1, при этом используются параметры изображения и значение громкости, сохраненные в памяти настроек.

Память каналов телевизора хранит до 60 каналов. Каналы нумеруются, начиная с единицы. Последовательное переключение каналов осуществляется при помощи кнопок «-» и «+». Нажатие на кнопку «+» позволяет переключить телевизор на канал с номером, на единицу большим (с 60-го канала телевизор переключается на 1-ый). Нажатие на кнопку «-» соответственно переключает телевизор на канал с номером, на единицу меньшим.

При нажатии на кнопку начальной установки очищается память каналов телевизора, после чего осуществляется поиск передач и сохранение их частот в памяти каналов. Поиск начинается с нижней границы рабочего диапазона телевизора. На экран телевизора выводится «синий экран». Рабочая частота постепенно увеличивается до тех пор, пока приемник не обнаружит телевизионный сигнал. Найденная передача выводится на экран в течение 10 секунд. Также отображается номер, под которым найденный канал будет сохранен в памяти (начиная с 1). Затем поиск продолжается до тех пор, пока не будет достигнута верхняя граница

диапазона, или пока не заполнена вся память каналов. Телевизор принимает управляющие сигналы с пульта ДУ. На пульте ДУ расположены следующие кнопки:

- кнопки с цифрами «0»...«9» для прямого переключения канала;
- кнопки «П-» и «П+» для последовательного переключения каналов;
- кнопки «Г-» и «Г+» для изменения громкости;
- кнопки «МЕНЮ», «←» и «→» для доступа к экранному меню.

Для прямого переключения на нужный канал его номер набирается с помощью кнопок с цифрами. После нажатия первой цифры в течение 5 секунд ожидается нажатие второй. Если вторая цифра не была нажата, то номер канала считается состоящим из одной цифры и осуществляется переключение на него.

Кнопки «П-» н «П+» на пульте имеют те же функции, что и кнопки «-» и «+» на корпусе телевизора. Кнопки «Г-» и «Г+» позволяют увеличивать или уменьшать громкость. Каждое нажатие на «Г-» уменьшает громкость на одну единицу, пока она больше нуля, «Г+» увеличивает громкость на единицу, пока не достигнуто максимальное значение. Текущее значение громкости сохраняется в памяти настроек.

Кнопки «МЕНЮ», «←» и «→» позволяют устанавливать значения настоек, хранящихся в памяти телевизора. При нажатии на кнопку «МЕНЮ» внизу экрана возникает надпись «ЯРКОСТЬ» и полоса, отображающая текущее значение яркости. Кнопками «←» и «→» яркость можно уменьшить или увеличить. При работе с меню нажатия на все остальные кнопки игнорируются. После повторного нажатия на кнопку «МЕНЮ» значение яркости запоминается в памяти настроек, и осуществляется переход к настройке контрастности. Настройка контрастности и остальных параметров (четкости, цветовой гаммы) происходит аналогично. Нажатие на кнопку «МЕНЮ» по окончании настройки цветовой гаммы (последнего пункта меню) приводит к окончанию работы с меню. Выход из меню также осуществляется в том случае, если в течение 15 секунд не была нажата ни одна кнопка.

Вариант задания включает в себя разработку модели системы, обеспечивающую управление телевизором, хранение и изменение информации о каналах и настройках телевизора.

Вариант 20. Домофон

Тема: Объектно-ориентированный анализ и проектирование программного обеспечения встроенного микропроцессора домофона.

Требуется разработать средствами UMLмодель программного обеспечения встроенного микропроцессора домофона.

Домофон регулирует доступ в подъезд многоквартирного дома. В подъезде имеется дверь с замком. С наружной стороны двери установлена панель с кнопками на каждую квартиру, микрофон и динамик. В каждой квартире имеется кнопка «СВЯЗЬ», «БЛОКИРОВКА» и «ОТКРЫТЬ». Кроме того, в квартире имеется микрофон и динамик.

Жильцы могут открывать дверь ключом. Посетитель может нажать кнопку квартиры на внешней панели. При этом в квартире раздается звонок (если подача звонка в квартиру не заблокирована). Услышав тонок, жилец квартиры нажимает на кнопку «СВЯЗЬ» внутренней панели домофона, после чего домофон устанавливает звуковое сообщение между жильцом и посетителем. Звуки, произносимые посетителем в микрофон, установленный на внешней панели, воспроизводятся в динамике, установленном в квартире. Звуки из микрофона в квартире, передаются в динамик на внешней панели. После сеанса связи жилец может нажать на кнопку «ОТКРЫТЬ», чтобы замок на двери в подъезд открылся, и посетитель смог войти. По истечении минуты замок должен снова заблокировать вход в подъезд. Жилец, который желает, чтобы его не беспокоили, может отключить подачу звонка в свою квартиру, нажав на кнопку «БЛОКИРОВКА». Повторное нажатие на эту кнопку вновь включает подачу звонка.

Вариант 21. Система поддержки составления расписания занятий

Тема: Объектно-ориентированный анализ и проектирование программного средства, обеспечивающего поддержку составления расписания занятий.

Требуется разработать средствами UMLмодель системы поддержки составления расписания занятий.

Система обеспечивает составление расписания некоторого учебного заведения, внесение в расписание изменений, выдачу полного расписания и дополнительной информации (например, по итоговому расписанию составляется расписание указанной группы на заданный день или неделю).

В расписании фиксируются время и место проведения занятия, предмет и преподаватель, проводящий занятие, а также номер группы, для которой это занятие проводится. Расписание не должно содержать коллизий (например, разные занятия не должны пересекаться друг с другом по месту и времени их проведения, один преподаватель не может вести одновременно два разных занятия, в одно и то же время у одной и той же группы не может быть два различных занятия и т. д.). При работе над этим вариантом задания необходимо разработать схему базы данных для хранения и изменения информации об учебных группах, преподавателях и предметах, а

также обеспечить процесс составления расписания.

Критерии оценки:

Максимально можно получить 40 баллов.
За каждую диаграмму UMLможно получить от 0 до 4 баллов, в зависимости от правильности выполнения. Всего 8 * 4 = 32 балла.
Отчёт по проделанной работе оценивается от 0 до 5 баллов.
Защита работы оценивается от 0 до 3 баллов.

ДОКЛАД

Задание: подготовить доклад по предложенной тематике.

Примерная тематика докладов:

1. Методология функционального моделирования SADT. Диаграммы потоков данных DFD. Диаграмма «сущность-связь». Объектно-ориентированный подход в моделировании систем.
2. UML — язык универсального моделирования. Диаграммы UML.
3. Обзор современных средств автоматизированного проектирования информационных систем.
4. Обзор стандартных механизмов доступа к данным – Borland Database Engine (BDE), Open Database Connectivity (ODBC), OLEDB, Active XDataObjects (ADO), JavaDataBaseConnectivity (JDBC).
5. Особенности универсального механизма доступа к данным UniversalDataAccess как стратегия предоставления доступа к любому типу информации.
6. Стандарт и реализация языка SQL. Формы и типы данных SQL. Язык определения данных (DDL). Язык манипулирования данными (DML).
7. Технология клиент-сервер. Модели клиент-сервер. Этапы развития серверов баз данных.
8. Инструментальные средства разработки приложений для информационных систем.
9. Географические информационные системы (ГИС). Особенности ГИС «MapInfo».
10. Системы автоматизированного проектирования САПР(CAD). Отечественные САПР «nanoCAD» и «КОМПАС».
11. Локальные и корпоративные информационные системы управления.
12. Системы поддержки принятия решений. МетодологияOLAP (On-lineAnaliticalProcessing).
13. Методологии создания экономических информационных управляющих систем(MRP,CRP, MRP II. ERP,. BPM).
14. Инструментальные средства осуществления технологических процессов сбора, размещения, хранения, накопления, преобразования и передачи информации.
15. Модели восстановления данных. Технологии и инструменты экспортирования данных и восстановления информации.
16. Обеспечение безопасности информационной системы. Средства для осуществления атак, системного и сетевого аудита.
17. Защита персональных данных. Средства защиты государственных информационных систем.

Критерии оценки доклада

Оценки в баллах	Критерии
2	Самостоятельность в подготовке материала
3	Аргументированность и полнота материала
2	Использование современной и актуальной литературы
3	Качество презентации
10	**Итого**