МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе
(ТИС (филиал) ДГТУ в г.Ставрополе)

УТВЕРЖДАЮ
Директор
_____ Е.А. Дрофа
20.04 2022 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ по дисциплинам
для студентов направления подготовки
09.04.02 Информационные системы и технологии
программа магистратуры «Информационные системы и технологии»

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

УТВЕРЖДАЮ
Директор
_____Е.А. Дрофа
_____ 2022 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению практических работ по дисциплинам
для студентов направления подготовки
09.04.02 Информационные системы и технологии
программа магистратуры «Информационные системы и технологии»

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Специальные главы математики» для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине « Специальные главы математики» содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии направленность (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины формирование у обучающихся компетенций, предусмотренных ФГОС ВО, что достигается в процессе формирования у обучающихся четкого мировоззрения о естественно-научной картине мира на основе понятий, законов и теорий современной и классической физики; формирования представлений о методологии науки на примере классической и современной экспериментальной и теоретической физики; адаптации обучающихся к восприятию материала учебных дисциплин, базирующихся на физических принципах, законах, явлениях и моделях.

- обучение приёмам исследования и решения математически формализованных задач; выработки у обучающихся умения анализировать полученные результаты; привитием навыков самостоятельного изучения литературы по математике и её приложениям;

- формированием мировоззрения и развитию системного мышления.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

ОПК-1.1: Анализирует естественнонаучные и общеинженерные знания, используемые при конструировании изделий легкой промышленности;

Изучив данный курс, студент должен:

Знать:

основные принципы и математические методы анализа и оптимизации управленческих решений.

Уметь:

выбирать рациональные варианты действий в практических задачах принятия решений с использованием экономико-математических моделей.

Владеть:

- методикой использования математической символики для выражения отношения объектов;

- методами дифференциального и интегрального исчислений;

- методами аналитического решения дифференциальных уравнений;

- основными алгоритмами моделирования процессов на базе линейной алгебры, аналитической геометрии и математического анализа в экспериментальных исследованиях в области техносферной безопасности.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических и практических занятиях. Лабораторные работы имеют целью углубить и закрепить полученные знания на лекциях и практических занятиях, практическое освоение обучаемыми научно-теоретических положений дисциплины, овладение понятийным аппаратом по изучаемым разделам (темам), методами экспериментальных и научных исследований, привитие навыков научного анализа и обобщения полученных результатов, навыков работы лабораторным оборудованием, контрольно-измерительными приборами и навыков вычисления погрешностей результатов измерений. Обязательным элементом в начале выполнения практических работ является инструктаж студентов по мерам безопасности. Лабораторная работа состоит из следующих этапов: доведение целей и решаемых задач каждой лабораторной работы; инструктаж обучающихся по мерам безопасности ,

проверка преподавателем подготовленности студентов и их допуск к выполнению работы; выполнение обучающимися экспериментального исследования под контролем преподавателя (лаборанта); оформление студентами результатов работы и формулирование выводов; защита отчетов по лабораторной работе.

Практическое занятие 1 Математическое моделирование и оптимизация технологических процессов.

Цель занятия заключается в формировании у студентов общепрофессиональной компетенции: ОПК-1.1

Задачи для обсуждения и задания по теме: «Математическое моделирование и оптимизация технологических процессов.

»

Задача 1. в пространстве трёх товаров рассмотрите бюджетное множество при векторе цен p= (2; 5; 6) и доходе q=30. описать его и его границу с помощью обычных и векторных неравенств и равенств, изобразить бюджетное множество и его границу графически. в ответе дайте число, равное объёму бюджетного множества.

задача 2. для потребителя с функцией полезности найдите в общем виде функцию спроса на оба товара при заданных ценах p =(2;5) и доходе q=40.

Практическое занятие 2 Методы линейной алгебры в моделировании социальных и технологических процессов.

Цель занятия заключается в формировании у студентов общепрофессиональной компетенции: ОПК-1.1

Задачи для обсуждения и задания по теме: «Методы линейной алгебры в моделировании социальных и технологических процессов»

Задача 1. В некоторой отрасли m заводов выпускают n видов продукции. Матрица $A_{m\times n}$ задаёт объёмы продукции на каждом заводе в первом квартале, матрица $B_{m\times n}$ - соответственно во втором; (aij, bij) – объёмы продукции j –го типа на i –м заводе в 1-м и 2-м кварталах соответственно:

$$A=\begin{pmatrix}2 & 3 & 7\\ 1 & 2 & 2\\ 4 & 1 & 5\\ 2 & 1 & 3\end{pmatrix};\quad B=\begin{pmatrix}3 & 0 & 2\\ 2 & 4 & 1\\ 4 & 3 & 2\\ 5 & 2 & 4\end{pmatrix}.$$

Найти:

а) объёмы продукции;

б) прирост объёмов производства во втором квартале по сравнению с первым по видам продукции и заводам;

в) стоимостное выражение выпущенной продукции за полгода (в долларах), если

λ – курс доллара по отношению к рублю.

Задача 2. Предприятие производит n типов продукции, используя m видов ресурсов. Нормы затрат ресурса i-го товара на производство единицы продукции j-го типа

заданы матрицей затрат $A_{m\times n}$. Пусть за определённый отрезок времени предприятие выпустило количество продукции каждого типа x_{ij}, записанное матрицей $X_{n\times 1}$.

Определить S – матрицу полных затрат ресурсов каждого вида на производство всей продукции за данный период времени, если

$$A_{4\times 3}=\begin{pmatrix}2 & 5 & 3\\ 0 & 1 & 8\\ 1 & 3 & 1\\ 2 & 2 & 3\end{pmatrix},\quad X_{3\times 1}=\begin{pmatrix}100\\ 80\\ 110\end{pmatrix}.$$

Решение. Матрица полных затрат ресурсов S определяется как произведение матриц, т.е. S=AX.

$$S=\begin{pmatrix}2 & 5 & 3\\ 0 & 1 & 8\\ 1 & 3 & 1\\ 2 & 2 & 3\end{pmatrix}\cdot\begin{pmatrix}100\\ 80\\ 110\end{pmatrix}=\begin{pmatrix}930\\ 960\\ 450\\ 690\end{pmatrix},$$

т.е за данный период времени будет израсходовано 930 ед. ресурса 1-го вида, 960 ед. ресурса 2-го вида, 450 ед. ресурса 3-го вида, 630 ед. ресурса 4-го вида.

Задача 3. Завод производит двигатели, которые могут либо сразу потребовать дополнительной регулировки (в 40% случаев), либо сразу могут быть использованы (в 60% случаев). Как показывают статистические исследования, те двигатели, которые изначально требовали регулировки, потребуют дополнительной регулировки через месяц в 65% случаев, а в 35% случаев через месяц будут работать хорошо. Те же двигатели, которые не требовали первоначальной регулировки, потребуют её через месяц в 20% случаев и продолжат хорошо работать в 80% случаев. Какова доля двигателей, которые будут работать хорошо или потребуют регулировки через 2 месяца после выпуска? Через 3 месяца?

Практическое занятие 3 Геометрия многомерного линейного пространства.

Цель занятия заключается в формировании у студентов общепрофессиональной компетенции: ОПК-1.1

Задачи для обсуждения и задания по теме: «Геометрия многомерного линейного пространства. »

Задача 1. Для изготовления изделий двух видов склад может отпустить материала не более 80 метров, причем на изделие I вида расходуется 2 метра, а на изделие II вида - 1 метр материала. Требуется спланировать производство так, чтобы была обеспечена наибольшая прибыль, если известно, что изделий I вида требуется не более 30 шт., а изделий II вида не более 40 шт., причем одно изделие I вида стоит 5 у.е., а II вида - 3 у.е.

Задача 2. На текстильной фабрике выпускаются покрывала и пледы. Производство устроено так, что вместо двух пледов, можно выпустить одно покрывало, причем покрывало приносит в 1,5 раза 6ольше прибыли, чем плед. Фабрика может произвести 700 изделий в день, однако склад может принять не более 500 штук в день. Сколько нужно выпустить в день покрывал и пледов, что бы фабрика получала максимальную прибыль?

Задача 3. Предприятие имеет 3 типа обрабатывающих станков A, B, C, на которых изготовляются изделия вида 1 и 2. Изделия первого вида вырабатываются на станках A и C, а изделия второго — на станках всех трех видов, т.е. A, B и C. Производственная мощность станков отдельных типов такова:

Тип станка	Производительная мощность (тыс. штук в год)
A	6 изделий 1 или 6 изделий 2 вида
B	4 изделия 2 вида
C	5 изделий 1 или 10 изделий 2 вида

Прибыль на единицу изделия 1 составляет 2 усл. ед., на ед. изд. 2 - 4 усл. ед. Определить такие объемы производства изделия 1 и 2, чтобы предприятие получило максимальную прибыль.

Практическое занятие 4 Анализ чувствительности оптимального решения к параметрам задачи линейного программирования.

Цель занятия заключается в формировании у студентов общепрофессиональной компетенции: ОПК-1.1

Задачи для обсуждения и задания по теме: «Анализ чувствительности оптимального решения к параметрам задачи линейного программирования. »

Задача 1. На промышленном предприятии изготавливаются два продукта: 1 и 2. Эта продукция производится с помощью оборудование u1, u2, u3, которое в течение дня может работать 24000, 27000 и 40000 сек. Норму времени, необходимого для производства ед. продукции с помощью соответственного оборудования, приводится в таблице:

Изделие	Оборудование		
	u1	u2	u3
1	3	8	9
2	6	4	3
Время работы в течение дня	24000	40000	27000

Прибыль от производства изделия 1 составляет 9 ед., а изделия 2 - 6 ед. Найти такой объем производства, чтобы прибыль была максимальной. (2400 сек. = 24 тыс. сек. и т.д.)

Задача 2. Производственный цех деревообрабатывающей промышленности ежемесячно имеет в своем распоряжении 48 м3 пиломатериалов и 45 м2 стекла. В цехе изготавливают два вида шкафов: конторские и библиотечные. Расход материалов на один шкаф каждого вида приведен в таблице. Сбытовая цена конторского шкафа 4000 усл. ед. Определить такой ассортимент, при котором месячный доход будет максимальным.

Вид шкафа	Сырье	
	Пиломатериалы	Стекло
Конторские	0,3	0
Библиотечные	0,3	1,5
Запасы	48	45

Задача 3. Завод заготавливает 2 вида изделий на экспорт с помощью машин u1 и u2. Максимальное время работы машин u1 8 часов, а машины u2 - 12 часов в сутки. Расход времени работы машин (в сутки) представлен в таблице (в час). Валютная прибыль от продажи единицы изделия 1 составляет 3 доллара, а изделия 2 - 4 доллара. Рассчитать производственный план на сутки при максимальной валютной прибыли.

Изделие	Машины	
	u1	u2
1	1	2,5
2	4	2
	8	12

Практическое занятие 4 Применение принципа гарантированного результата в задачах экономического планирования.

Цель занятия заключается в формировании у студентов общепрофессиональной компетенции: ОПК-1.1

Задачи для обсуждения и задания по теме: «Применение принципа гарантированного результата в задачах экономического планирования.»

Задача 1. Издержки y (в руб.) на изготовление партии деталей определяются по формуле $y = ax + b$, где x - объём партии. Для первого варианта технологического процесса $y = 1{,}45x + 20$. Для второго варианта известно, что $y = 157{,}5$ (руб.) при $x = 100$ (дет.) и $y = 452{,}5$ (руб.) при $x = 300$ (дет.). Провести оценку двух вариантов технологического процесса и найти себестоимость продукции для обоих вариантов при $x = 200$ (дет.)

Решение.

Для второго варианта определяем параметры a и b из системы уравнений:

$\begin{cases} 157{,}5 = a \cdot 100 + b, \\ 425{,}5 = a \cdot 300 + b, \end{cases}$ откуда $a = 1{,}475$ и $b = 10$, т.е. $y = 1{,}475 \cdot x + 10$.

Точка (x0,y0) пересечения двух прямых находится из системы их уравнений:

$\begin{cases} y = 1{,}45x + 20, \\ y = 1{,}475x + 10, \end{cases}$ откуда $x_0 = 400$, $y_0 = 600$.Очевидно, при объёме партии $x \prec 400$ выгоднее второй вариант технологического процесса, при $x \succ 400$ - первый вариант. Себестоимость продукции (руб.) при $x = 200$ по первому варианту составляет $y = 1{,}45 \cdot 200 + 20 = 310$, а по второму - $y = 1{,}475 \cdot 200 + 10 = 305$.

Задача 2. Постоянные издержки F составляют 125 тыс.руб. в месяц, а переменные издержки $V(x)$ - 700 руб. за каждую единицу продукции. Цена единицы продукции 1200 руб. Найти объём продукции x, при котором прибыль равна: а) нулю (точка безубыточности); б) 105 тыс.руб. в месяц.

Решение:

а) Издержки производства x единиц продукции составят: $C(x) = F + V(x) = 125 + 0{,}7x$ (тыс.руб.). Совокупный доход (выручка) от реализации этой продукции $R(x) = 1{,}2x$, а прибыль $P(x) = R(x) - C(x) = 0{,}5x - 125$ (тыс.руб.). Точка безубыточности, в которой $P(x) = 0{,}5x - 125 = 0$, равна $x = 250$ (ед.).

б) Прибыль $P(x) = 105$ (тыс.руб.), т.е. $P(x) = 0{,}5x - 125 = 105$ при $x = 460$ (ед.).

Задача 3. Продолжительность выполнения y (мин.) при повторных операциях связана с числом x этих операций зависимостью $y = \dfrac{a}{x + c}$. Вычислить, сколько минут выполняется работа при 50 операциях, если известно, что при $x = 20$ $y = 125$, а при $x = 200$ $y = 50$.

Решение. Найдём параметры a и c, учитывая, что $y(20) = 125$, $y(200) = 50$. Получаем систему: $\begin{cases} 125 = \dfrac{a}{20 + c}, \\ 50 = \dfrac{a}{200 + c}, \end{cases}$ решая которую найдём $a = 15000$, $c = 100$.

Итак, $y = \dfrac{15000}{x + 100}$ при $x = 50$, $y(50) = \dfrac{15000}{50 + 100} = 100$ (мин.)

Практическое занятие 6 Методы предельного анализа в производственных процессах.

Цель занятия заключается в формировании у студентов общепрофессиональной компетенции: ОПК-1.1

Задачи для обсуждения и задания по теме: «Методы предельного анализа в производственных процессах.

»

Задача 1. выяснить, чему равны предельные и средние полные затраты предприятия, если эластичность полных затрат равна 1?

Задача 2. для потребителя с функцией полезности найдите в общем виде функцию спроса на оба товара при заданных ценах p =(2;5) и доходе q=40.

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

Основная литература				
	Авторы, составители	Заглавие	Издательство	Адре
Л1.1	Шипачев В. С.	Высшая математика: Учебник	Москва: ООО "Научно-издательский центр ИНФРА-М",	http://znanium.com/go.php?

Л1.2	Шепелева, Р. П., Головко, Н. И., Иванов, Б. Н., Первухин, М. А., Полещук, Г. С., Коробецкая, Ю. И., Бондрова, О. В., Крылова, Д. С.	Математика: учебное пособие	Саратов: Ай Пи Эр Медиа, 2018	http://www.iprbookshop.ru/70267.html
. Дополнительная литература				
	Авторы, составители	Заглавие	Издательство	Адре
Л2.1	Дадаян А. А.	Математика: Учебник	Москва: Издательство "ФОРУМ", 2010	http://znanium.com/go.php?
Методические разработки				
	Авторы, составители	Заглавие	Издательство	Адре
Л3.1	Темирова, Л. Г.	Базы данных: учебно-методическое пособие для выполнения практических работ для студентов iii курса обучающихся по направлению подготовки 231300.62 прикладная математика	Черкесск: Северо-Кавказская государственная гуманитарно-	http://www.iprbookshop.ru/2717
Перечень ресурсов информационно-телекоммуникационной сети "Интернет"				
Э1	Математика.: Учебник / А.А. Дадаян. - 3-е изд. - М.: Форум, 2010. - 544 с.: 60x90 1/16. - (Профессиональное образование). (переплет) ISBN 978-5-91134-460-3 - Режим доступа: http://znanium.com/catalog/product/242366			
Э2	Высшая математика: Учебник / Шипачев В.С. - М.:НИЦ ИНФРА-М, 2015. - 479 с.: 60x90 1/16 (Переплёт 7БЦ) ISBN 978-5-16-010072-2 - Режим доступа: http://znanium.com/catalog/product/469720			
Э3	Катрахова, А. А. Спецглавы математики и их приложения к задачам электромеханики и теории управления : курс лекций / А. А. Катрахова, В. С. Купцов, Е. М. Васильев. — Воронеж : Воронежский государственный технический университет, ЭБС АСВ, 2019. — 269 с. — ISBN 978-5-7731-0802-3. — Текст : электронный // Электронно- библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/93340.html			
Э4	Практикум по спецглавам высшей математики (ТФКП, ОИ, ТП) : учебное пособие / В. Я. Долгих, В. И. Бутырин, Г. В. Недогибченко, Э. Б. Шварц. — Новосибирск : Новосибирский государственный технический университет, 2014. — 97 с. — ISBN 978-5-7782-2499-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/45427.html			

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Математические модели информационных процессов» для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Математические модели информационных процессов» содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии, направленность (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины – освоение методов разработки математических моделей информационных процессов и методологии и технологии математического моделирования при исследовании, проектировании, эксплуатации информационных систем; формирование общекультурных и профессиональных компетенций магистра в соответствии с требованиями ФГОС по направлению Информационные системы и технологии; подготовка магистра к деятельности, требующей применение научно-практических знанийи умений в области анализи информационных процессов; развитие логического, алгоритмического мышления студентов, умения самостоятельно расширять свои знания в области математического представления информационных процессов.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

В результате освоения данной дисциплины формируется следующая компетенция у обучающегося:

ОПК-6.2: Оценивает процессы получения, передачи, хранения и представления информации на основе положений системной инженерии.

Изучив данный курс, студент должен:

Знать:

Инструментальные средства математического моделирования информационных процессов и условия их применимости.

Теоретические основы функционирования информационных систем и процессов и математический аппарат для их описания.

Уметь:

Создавать и исследовать математические модели информационных процессов с использованием стандартных пакетов автоматизированного проектирвания.

Владеть:

Созданием математических моделей информаационного процесса на языке высокого уровня и с использованием интегрированных пакетов прикладных программ.

Оценки адекватности математическкой модели.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическое занятие №1 Создание модели непрерывно-детерминированной системы

Целью практического занятия является практическое освоение методов моделирования непрерывно-детерминированных динамических систем. В процессе выполнения задания студенты должны:

разработать модели заданной системы, реализованные в пакете визуального моделирования Simulink на основе полученного описания;

получить результаты исследования данных моделей и выполнить их объяснение в терминах прикладной области.

В качестве примера учебного задания рассматривается задача об управлении протеканием воды через систему, состоящую из двух цилиндрических баков, расположенных на разной высоте, или, более кратко, задача о двух баках. Выбор этой задачи в качестве примера обусловлен следующими причинами:

понятным физическим принципом;

нелинейным поведением компонент системы;

наличием аварийных ситуаций в поведении системы, которые необходимо

обрабатывать;

1. Содержание задачи о двух баках

Система представляет собой два цилиндрических бака, расположенных вертикально на разной высоте таким образом, что дно первого бака находится на расстоянии H=0.39[м] от дна второго (рис.1). Баки имеют одинаковую высоту h=1[м] и различные диаметры: первый - D_1=12 [см], второй - D_2=5 [см]. Система имеет входную трубу, находящуюся в первом баке на расстоянии h от его дна. Баки соединены, трубой, являющейся выходной трубой первого бака (и располагающейся у самого его дна) и входной трубой второго бака (располагающейся на расстоянии H от его дна). Также система имеет выходную трубу, располагающуюся у самого дна второго бака. Входная труба системы снабжена входным краном V_{input} (он же входной кран первого бака). Труба, соединяющая выход первого бака со входом второго, снабжена краном V_1. Выходная труба системы снабжена выходным краном V_2 (он же выходной кран второго бака). Подача воды в систему контролируется краном V_{input} , который открывается мгновенно и скорость входного потока воды определяется как (л/час):

$$\frac{dV_{input}}{dt} = \begin{cases} 0, если\ V_{input}\ закрыт, \\ 400\ , если\ V_{input}\ открыт \end{cases} \quad (1)$$

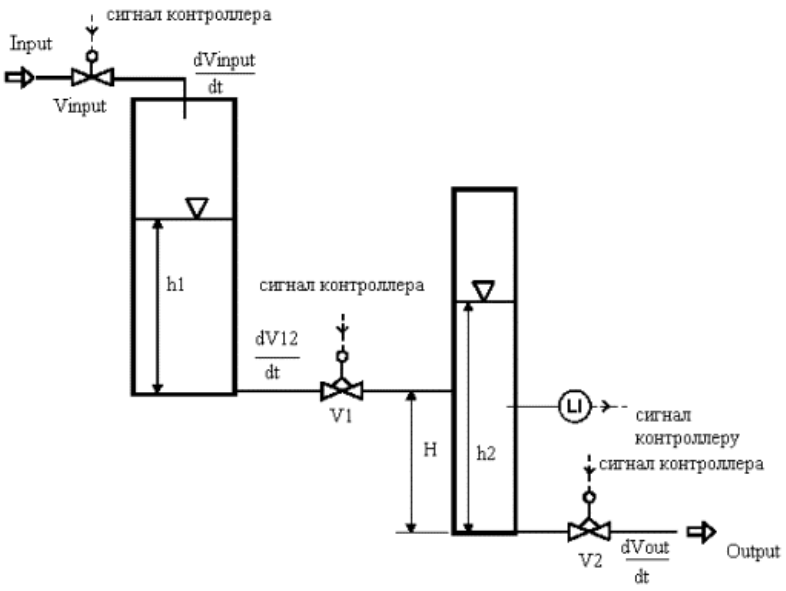


Рисунок 1 Система двух баков

Краны V_1 и V_2 являются медленными устройствами и открываются и закрываются с одной о той же постоянной скоростью, так что от момента начала открытия (закрытия) до полного открытия (закрытия) требуется 80 сек. Их открытие и закрытие контролируется задвижкой, меняющей свое положение от значения P=0 (полное закрытие в условных единицах) до P=80 (полное открытие).

Управление открытием/закрытием кранов $V_{input,}$ V1 и V2 осуществляется неким устройством, называемым контроллером и также входящим в описываемую систему.

Если через A_1 и A_2 обозначить площади оснований баков, то система уравнений для уровней воды в баках h_1 и h_2 запишется так:

$$\frac{dh_1}{dt} = \frac{1}{A_1}\left(\frac{dV_{input}}{dt} - \frac{dV_{12}}{dt}\right); \quad \frac{dh_2}{dt} = \frac{1}{A_2}\left(\frac{dV_{12}}{dt} - \frac{dV_{out}}{dt}\right); \quad (2)$$

где dV_{12}/dt - скорость протекания воды по трубе между баками, а dV_{out}/dt – скорость вытекания воды из системы. Скорость протекания воды между баками зависит от уровней воды h_1 и h_2, значения H и положения задвижки P_1 в кране V_1 :

$$\frac{dV_{12}}{dt} = \begin{cases} K_1(P_1)\cdot\sqrt{h_1-(h_2-H)}, & h_2 > H \\ K_1(P_1)\cdot\sqrt{h_1}, & h_2 \le H \end{cases} \quad (3)$$

Скорость вытекания воды из системы зависит от уровня воды во втором баке h2 и положения задвижки P2 на кране V2:

$$\frac{dV_{out}}{dt} = K_2(P_2)\cdot\sqrt{h_2} \quad (4)$$

Индивидуальные свойства кранов определяются функциями:

$$K_1(P_1) = \begin{cases} 1.85 \cdot 10^{-4} e^{-6 \cdot 10^{-6} p_1^3}, 0 \le P_1 < 80, \\ 0, \qquad P_1 = 80. \end{cases} \quad (5)$$

$$K_2(P_2) = \begin{cases} 2.26 \cdot 10^{-4} e^{-5.7 \cdot 10^{-6} p_1^3}, 0 \le P_2 < 80, \\ 0, \qquad P_2 = 80. \end{cases} \quad (6)$$

Работа всей системы описывается следующим алгоритмом. В исходном состоянии все краны закрыты и оба бака пусты. В начальный момент контроллер посылает сигнал входному крану V_{input}, тот мгновенно открывается и в течении времени Time1 [сек] наполняется только первый бак. По истечении времени Time1 контроллер посылает команду открыть кран V1 и вода начинает поступать во второй бак. Второе состояние сохраняется на протяжении Time2 [сек]. По истечении времени Time2 начинает контролироваться положение крана V2. А именно, если контроллер обнаруживает, что уровень воды во втором баке опустился ниже значения L_minus [м], поступает команда закрыть выходной кран, если вода во втором баке превышает уровень L_plus [м] – выдается команда открыть выходной кран. Аварийными считаются ситуации, когда переполняется один из баков или происходит периодическое открытие и закрытие выходного крана. Нормальным режимом системы считается состояние, когда все краны открыты, и вода протекает через систему с постоянной скоростью.

В качестве тестов предлагается пять задач (табл.1):

Таблица 1

	Time1 [c]	Time2 [c]	L_plus [м]	L_minus [м]
1	90	20	0.94	0.16
2	70	30	0.94	0.16
3	70	20	0.94	0.16
4	60	25	0.9	0.30
5	70	26.85	0.94	0.16

Они соответствуют ситуациям:

1. переполнение первого бака;
2. переполнение второго бака;
3. нормальный режим;
4. периодическое открытие и закрытие выходного крана;
5. система находится на границе между нормальным состоянием и режимом переполнения второго бака, оставаясь в нормальном состоянии.

В примере задачи с двумя баками можно выделить один основной вариант использования - *"Режим нормальной работы системы"*, а также три аварийные

ситуации, а именно: *"Переполнение первого бака"*, *"Переполнение второго бака"* и *"Периодическое открытие/закрытие выходного крана"*.

"Режим нормальной работы системы" содержит следующие требования:

В начальный момент времени контроллер посылает системе двух баков сигнал открыть входной кран,

По истечении Time1 контроллер посылает системе двух баков сигнал открыть кран V1,

В течении 80 сек. происходит открытие крана V1,

По истечении Time2 контроллер посылает системе двух баков сигнал открыть кран V2,

В течении 80 сек. происходит открытие крана V2.

Далее система работает в нормальном режиме, то есть, вода протекает свободно через систему, при этом не происходит переполнения баков.

"Периодическое открытие/закрытие выходного крана" расширяет основной вариант использования и содержит следующие дополнительные требования:

Если после включения входного крана уровень воды во втором баке становиться ниже уровня L_minus, то контроллер посылает системе двух баков сигнал закрыть кран V2,

В течении 80 сек. происходит закрытие крана V2,

Если уровень воды во втором баке становится выше уровня L_plus, то контроллер посылает системе двух баков сигнал открыть кран V2;

В течении 80 сек. происходит открытие крана V2.

Далее цикл замыкается и система работает в режиме периодического открытия/закрытия выходного крана.

"Переполнение первого бака" также расширяет основной вариант использования и содержит следующие дополнительные требования:

Если в любой момент работы системы происходит переполнение первого бака, система блокируется (то есть, прекращается всякое движение воды в системе).

"Переполнение второго бака" также расширяет основной вариант использования и содержит следующие дополнительные требования:

Если в любой момент работы системы происходит переполнение второго бака, система блокируется (то есть, прекращается всякое движение воды в системе).

Диаграмма вариантов использования представлена на рис.2:

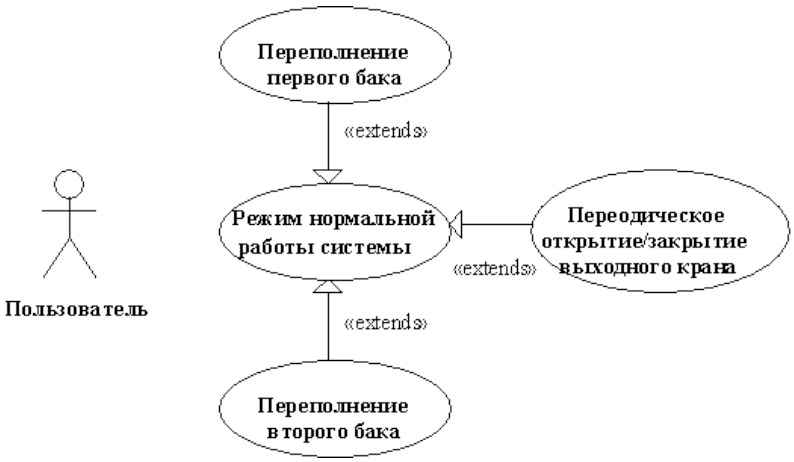


Рисунок 2 - Диаграмма вариантов использования

2. Модель системы, реализованная в подсистеме Simulink пакета Matlab

Построение модели в Simulink сводится к перемещению с помощью мыши необходимых блоков из библиотек Simulink в окно создаваемой модели и соединению этих блоков между собой с помощью функциональных связей. Библиотеки Simulink содержат большое количество разнообразных функциональных блоков, которые отображаются на экране в виде пиктограмм.

Используя описание задачи о двух баках можно выделить в данном примере подсистему System (рис.3), представляющую собой составной блок, который содержит в себе функциональную схему, содержащую в себе диаграмму Stateflow (представленную блоком Controller, являющимся экземпляром стандартного блока Chart и описывающую поведение контроллера) и составной блок Tank_System_Block, соединенные соответствующими функциональными связями (рис.4). Так же в блоке System присутствуют часы Clock, подающие системное время из Simulink в диаграмму Stateflow. Это объясняется тем, что при построении моделей, где используются блоки Stateflow (содержащие в себе переходы, инициируемые истечением неких временных интервалов) необходимо синхронизировать внутреннее системное время в Stateflow и в Simulink. Для этого необходимо на отдельный вход в блоке Stateflow подавать системное время из Simulink и именно это время использовать при составлении условий переходов (таких как истечение отрезков времени Time1 и Time2 в задаче о двух баках). Как видно из рис.4, к связи, соединяющей блок Clock и Controller, подсоединены два экземпляра блока Hit Cross. Еще два экземпляра блока Hit Cross подсоединены к связи блока Controller и выхода h2 блока Tank_System_Block. Использование Hit Cross блоков необходимо для правильного выполнения переходов в диаграмме Stateflow, включенной в модель Simulink, в которой происходит непрерывное интегрирование. На вход Hit Cross блока подается некая величина (в модели двух баков для двух Hit Cross блоков это системное время, для еще двух - величина h2). Сам Hit Cross блок содержит в себе некую величину, при совпадении которой с величиной, подаваемой на вход Hit Cross блока, система уменьшает шаг интегрирования. В связи со спецификой системы Stateflow это влияет на правильное выполнение переходов и не позволяет системе «проскочить момент», когда должен выполнится тот или иной переход. У первого Hit Cross блока внутренней величиной является момент времени Time1, у второго - Time 1+Time2, у третьего - значение L_plus, у четвертого - L_minus.

Блок System не имеет входов и имеет два выхода - h1 и h2, соединенные со стандартным блоком Mux (объединяющим их в вектор (h1,h2)), соединенный со стандартным блоком вывода Scope (на который он подает вектор выходных величин (h1,h2)).

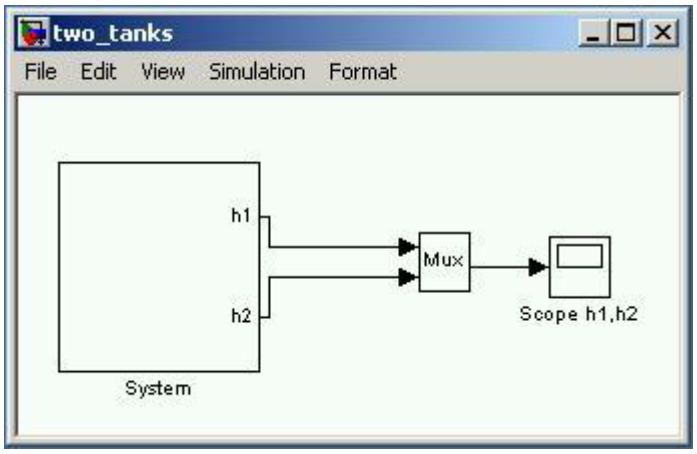


Рисунок 3

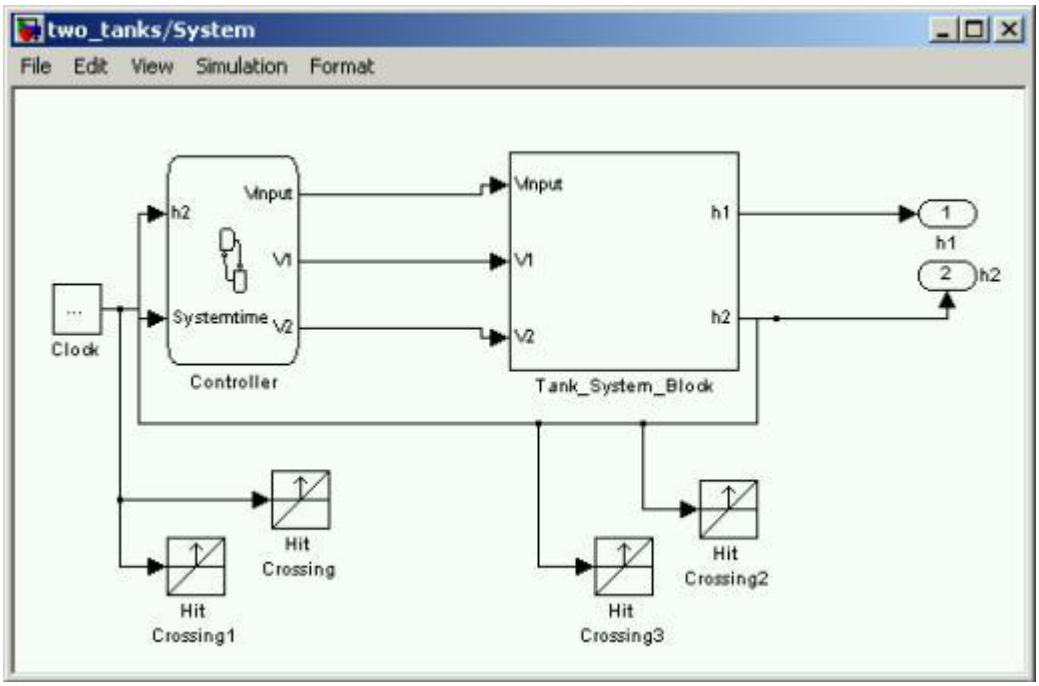


Рисунок 4

В свою очередь, составной блок Tank_System_Block содержит в себе функциональную схему, состоящую из составного блока Tank_System (имеющего три входа, соединенные с соответствующими выходами блока Chart, на которые подаются сигналы контроллера, и два выхода - h1 и h2) и пар экземпляров стандартных блоков Const, Relational Operetor и Stop Simulation (выполняющих проверку на переполнение баков), соединенных соответствующими функциональными связями (рис.5). Блок Tank_System_Block имеет три входа (на которые подаются сигналы от контроллера) и два выхода (h1 и h2).

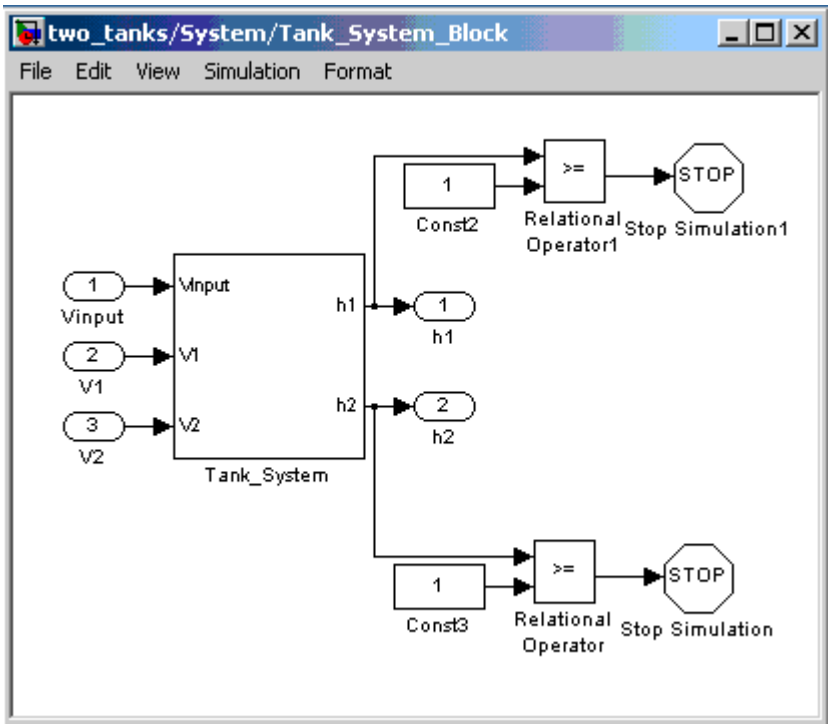


Рисунок 5

Составной блок Tank_System содержит в себе функциональную схему, состоящую из составных блоков Two_Tanks, Vin_Control, K1_Control и K2_Control, соединенных между собой функциональными связями (рис.6). Блок Tank_System имеет то же количество входов и выходов с теми же значениями, что и предыдущий блок.

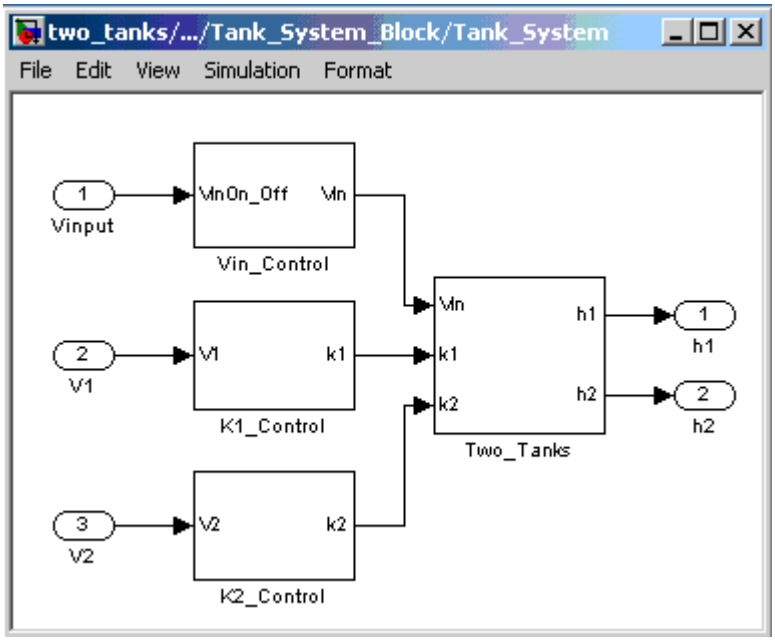


Рисунок 6

Составной блок Vin_Control содержит в себе функциональную схему, состоящую из стандартного блока Switch и двух экземпляров стандартного блока Const, содержащих в себе значения, соответствующие скорости входного потока воды в систему двух баков, когда входной кран открыт и закрыт. Переключения между этими значениями происходит в блоке Switch в зависимости от значения, поступающего на вход блока Vin_Control от

контроллера. Соответствующее значение Vin подается на выход. Блок Vin_Control имеет один вход и один выход (рис.7):

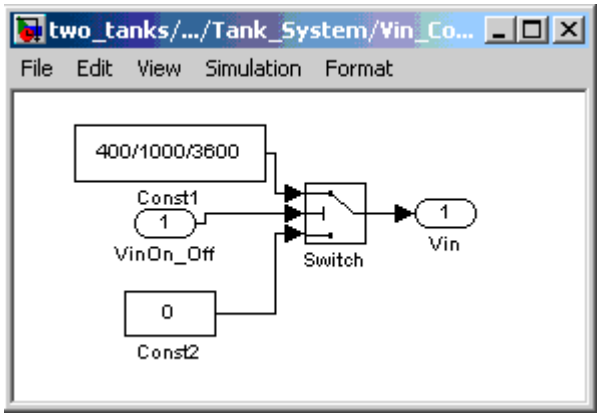


Рисунок 7

Составные блоки K1_Control и K2_Control содержат в себе идентичные функциональные схемы, различающиеся только значением коэффициентов в уравнении, представленном в стандартном блоке fcn. Функциональная схема состоит из двух экземпляров стандартного блока Switch (один из которых необходим для переключения между положениями кранов V1/V2 открыто/закрыто, а второй для отслеживания ситуации, когда параметр p становиться равным 80 и соответствующего переключения между значениями функции K(p)), трех экземпляров стандартного блока Const (содержащих в себе значения, 1 и -1, соответствующие положению крана V1/V2 (открывается/закрывается) и значение 0 для функции K(p) в ситуации, когда p>=80), стандартного блока Integrator (интегрирующего поступающее от переключателя значение в пределах от 0 до 80 с начальным значением интегрируемой величины 80) и стандартного блока f(u), в котором происходит вычисление значение функции K1(p)/K2(p). Переключения в блоке Switch происходит в зависимости от значения, поступающего на вход блока K1_Control/K2_Control от контроллера. Соответствующее значение k1/k2 подается на выход. Блок K1_Control/K2_Control имеет один вход и один выход (на рис.8 представлен блок K1_Control, блок K2_Control идентичен):

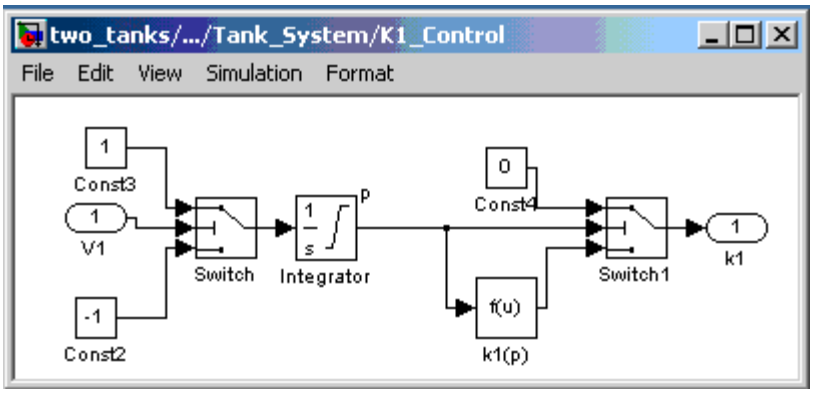


Рисунок 8

Составной блок Two_Tanks содержит в себе функциональную схему, состоящую из составных блоков Vout_calc, V12_calc1 и V12_calc2 (отвечающих за вычисление значений Vout и V12), стандартного блока Switch, двух экземпляров стандартного блока Mux (объединяющих в вектора соответствующие значения - Vin (поступающее на вход блока Two _Tanks с выхода блока Vin_Control)) и V12, а также V12 и Vout), двух экземпляров

стандартного блока fcn (производящих вычисление производных h1 и h2), двух экземпляров стандартного блока Integrator, выходами которых являются значения h1 и h2, поступающие на выход. Переключения в блоке Switch происходит в зависимости от значения h2, поступающего на вход блока Switch от интегратора, и в зависимости от этого значения, подключается один из блоков, вычисляющих V12. Блок Two_Tanks имеет три входа, на которые подаются значения Vin, k1 и k2; и два выхода - h1 и h2 (рис.9):

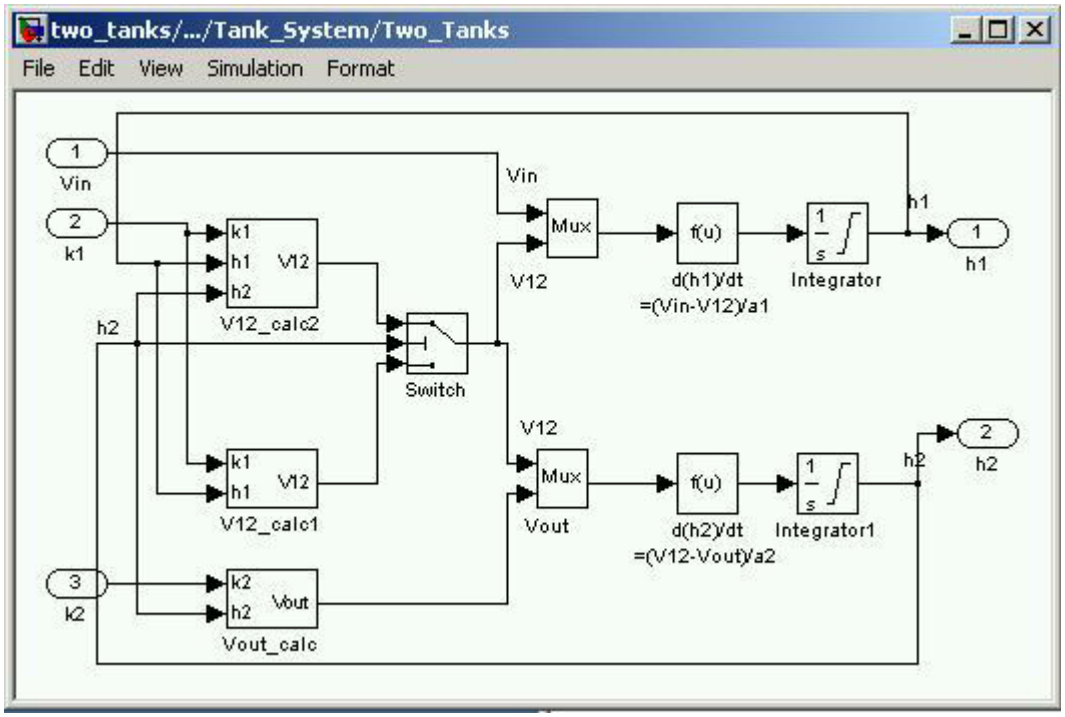


Рисунок 9

Составной блок Vout_calc содержит в себе функциональную схему, состоящую из стандартного блока Mux, объединяющего в один вектор значения k2 и h2, поступающие на вход блока Vout_calc, и подающего этот вектор на вход стандартного блока fcn, в котором происходит вычисление выходной величины Vout. Блок Vout_calc имеет два входа и один выход (рис.10):

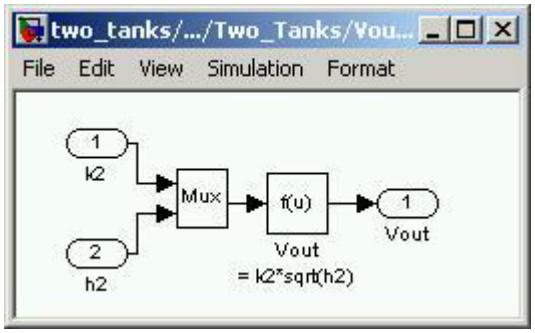


Рисунок 10

Составной блок V12_calc1 содержит в себе функциональную схему, состоящую из стандартного блока Mux, объединяющего в один вектор значения k1 и h1, поступающие на вход блока V12_calc1, и подающего этот вектор на вход стандартного блока fcn, в котором происходит вычисление выходной величины V12. Блок V12_calc1 имеет два входа и один выход (рис.11):

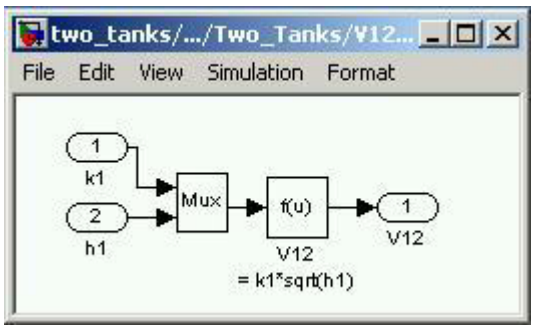


Рис. 11

Составной блок V12_calc2 содержит в себе функциональную схему, состоящую из стандартного блока Mux, объединяющего в один вектор значения k1, h1 и h2, поступающие на вход блока V12_calc2, и подающего этот вектор на вход стандартного блока fcn, в котором происходит вычисление выходной величины V12. Блок V12_calc2 имеет три входа и один выход (рис.12):

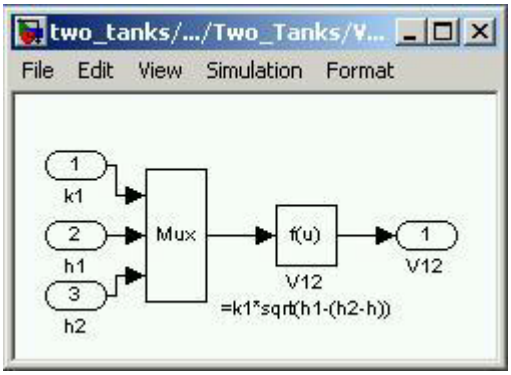


Рис. 12

Диаграмма Stateflow, представлена экземпляром стандартного блока Chart, (рис.13). В ней вместо внутреннего времени t используется подаваемое на вход блока системное время Simulink (входная переменная Systemtime).

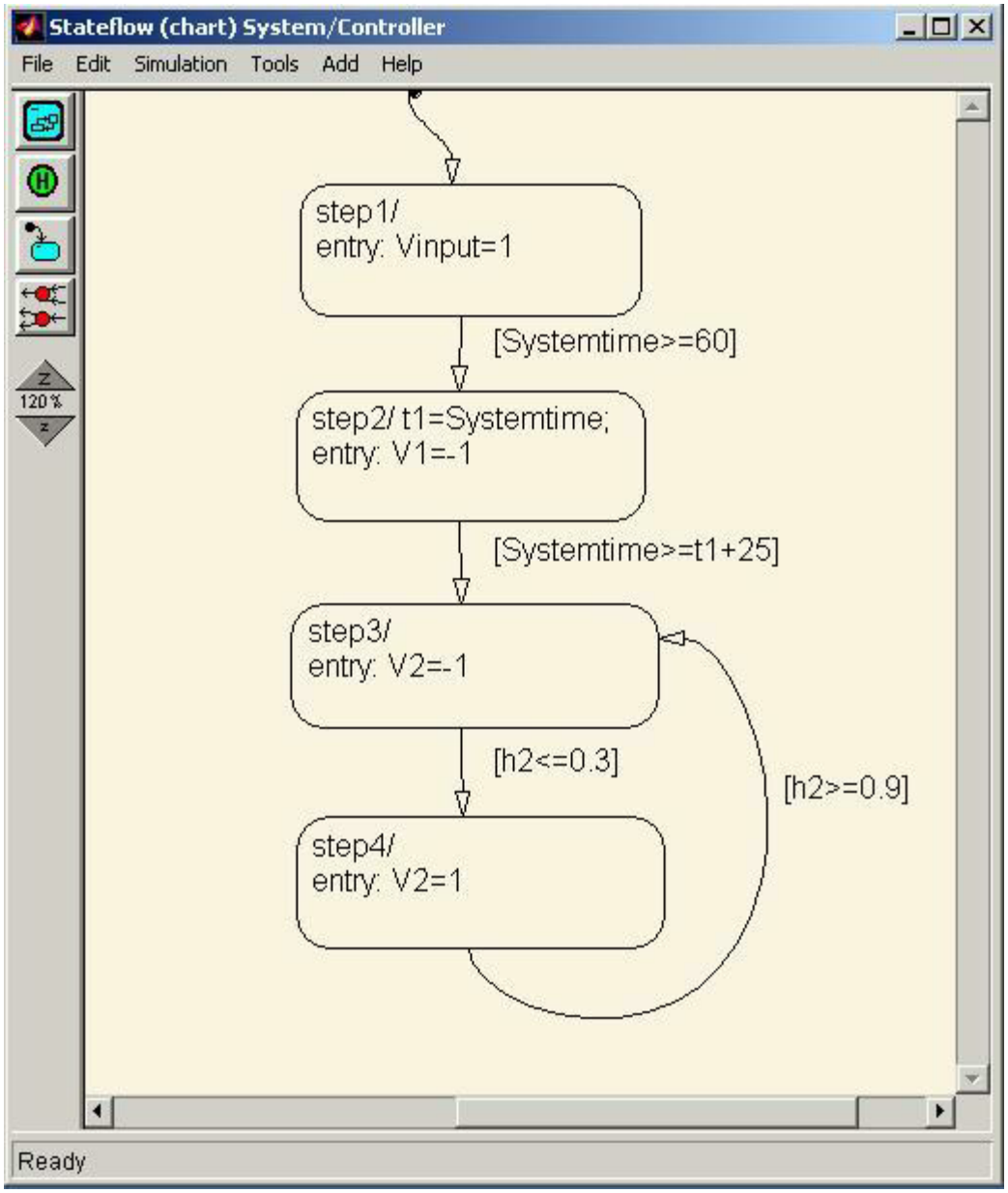


Рис. 13

Результаты эксперимента

На рис.14 представлена временная диаграмма изменений уровней воды в баках, полученная в результате эксперимента с моделью, реализованной в подсистеме Simulink пакета Matlab:

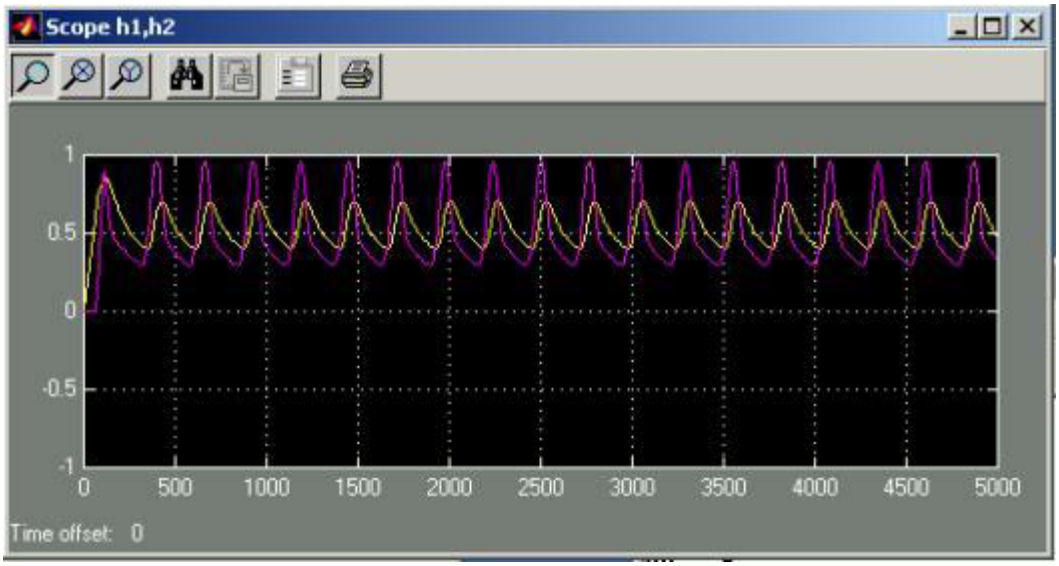


Рис. 14

Данные временные диаграммы отражают изменения уровней воды в системе двух баков при открытии/закрытии выходного крана. Возрастающие участки кривых $h_1(t)$ и $h_2(t)$ соответствуют состоянию системы, когда выходной кран закрыт. Убывающие участки кривых $h_1(t)$ и $h_2(t)$ соответствуют состоянию системы, когда выходной кран открыт.

Практическое занятие №2 Создание стохастической имитационной модели системы

Целью практического занятия является практическое освоение методов моделирования стохастических динамических систем. В процессе выполнения задания студенты должны:

разработать модель заданной системы с использованием пакета визуального моделирования Simulink на основе полученного описания;

получить результаты исследования модели и выполнить их объяснение в терминах прикладной области.

1. Описание моделируемой системы

В практике передачи и приема информации с использованием радиоканалов связи одной важнейших задач является оценка влияния помех на вероятность безошибочной передачи информации. Рассмотрим систему цифрового радиоканала, представленную на рис. 1.

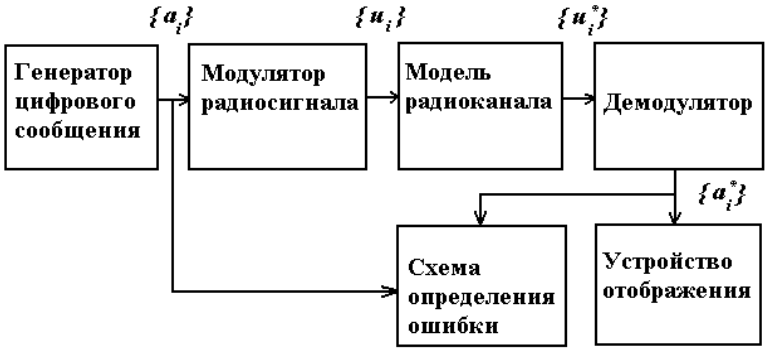


Рисунок 1

Генератор цифровых сообщений служит для имитации двоичных кодов заданной длины и может быть представлен следующим выражением

$$\begin{cases} a_i = 0, \textit{если } Rnd(i) < 0{,}5 \\ a_i = 1, \textit{если } Rnd(i) \geq 0{,}5 \end{cases},$$

где $Rnd(i)$ число из интервала 0…1, выдаваемое встроенным программным генератором псевдослучайных чисел; $i = \overline{1, I_{зад}}$.

Несущий радиосигнал задан выражением

$$u(t) = U \sin(2\pi f_n t + \varphi_0),$$

где U амплитуда радиосигнала; f_n - несущая частота; φ_0 - начальная фаза.

Модуляция радиосигнала по закону передаваемого сообщения может быть выполнена с помощью импульсной модуляции (ИМ):

$$\begin{cases} U = 0, \textit{если } a_i = 0; \\ U = 1, B, \textit{если } a_i = 1. \end{cases}$$

Частотной модуляции (ЧМ):

$$\begin{cases} f = f_n, \textit{если } a_i = 0; \\ f = f_n + \Delta f, \textit{если } a_i = 1. \end{cases}$$

Фазокодовой манипуляции (ФКМ):

$$\begin{cases} \varphi_0 = 0, \textit{если } a_i = 0; \\ \varphi_0 = \pi, \textit{если } a_i = 1. \end{cases}$$

Модель канала характеризуется видом помех. Для аддитивных помех, приводящих к искажению амплитуды сигнала, используется выражение

$$u^*(t) = u(t) \pm Rnd(t) \cdot U_П,$$

где $U_П$ - максимальное значение амплитуды помехи.

Модель канала с фазовыми искажениями имеет вид

$$u^*(t) = u(t) + U_П \cdot \sin(2\pi f_n t \pm Rnd(t) \cdot \varphi_0).$$

Модель канала с преобладающими частотными искажениями может быть задана выражением

$$u^*(t) = u(t) + U_П \cdot \sin(2\pi t(f_n \pm Rnd(t) \cdot \Delta f).$$

Демодуляция определяется видом модуляции радиосигнала и для амплитудной импульсной модуляции задается выражением

$$\begin{cases} u^*(t) > U^1_{порога}, то\ a_i^* = 1; \\ u^*(t) < U^0_{порога}, то\ a_i^* = 0, иначе\ a_i^* = x. \end{cases}$$

Пороговые значения нулевого и единичного уровней определяются исходя из условий распространения сигнала и дальности связи. Состояние *x* характеризует появление ошибки. Аналогичные выражения только с пороговыми значениями частоты или фазы сигнала используются для демодуляции ЧМ и ФКМ сигналов.

Фиксация ошибки осуществляется посредством сравнения $a_i = a_i^*$ или по наличию ошибки демодуляции $a_i^* = x$.

2. Задание для самостоятельной работы

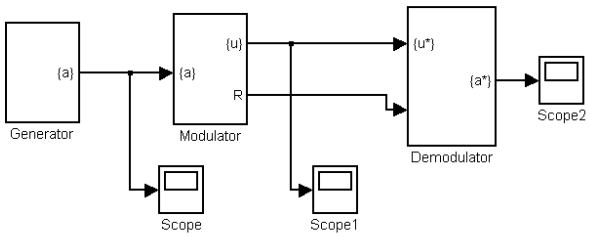
В соответствии с номером по журнальному списку выбрать из табл. 1 исходные данные для моделирования и составить модель радиоканала. При составлении модели максимально использовать возможности пакета имитационного моделирования Simulink. Оформить отчет по практическому занятию в котором отразить текст описания системы и разработанной модели. Привести графики на выходе элементов модели для различных вариантов исходных данных. Оценить влияние различных помеховых воздействий на число ошибок.

Таблица 1

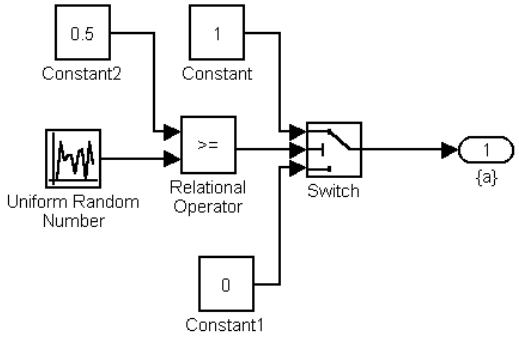
№ п/п	Исходные данные для моделирования					Примечание
	Вид модуляции	Число разрядов	$U^1_{порога}$	$U^0_{порога}$	$U_П$	
1	ИМ	7	$0{,}6U$	$0{,}1U$	$0{,}01U$	
2	ФКМ	8				
3	ЧМ	9				Δf =0,1 f

4	ИМ	10	$0{,}8U$	$0{,}2U$	$0{,}02\ U$	
5	ЧМ	7				$\Delta f = 0{,}2\ f$
6	ФКМ	8				
7	ЧМ	9				$\Delta f = 0{,}01\ f$
8	ИМ	10	$0{,}9U$	$0{,}1U$	$0{,}01\ U$	
9	ФКМ	7				
10	ИМ	8	$0{,}5U$	$0{,}05U$	$0{,}01\ U$	

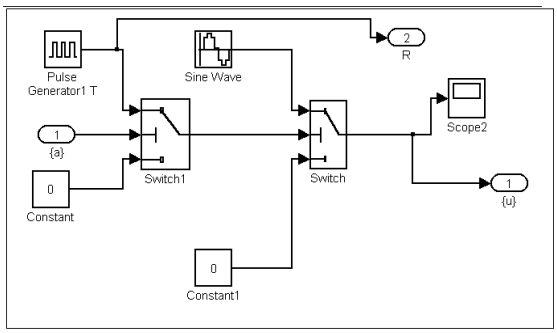
Структура модели



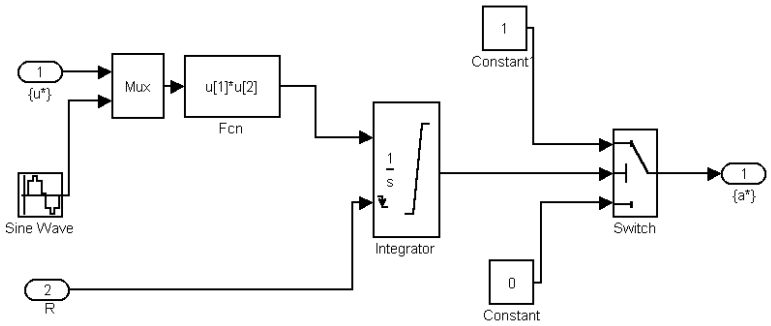
Модель генератора цифрового сигнала



Модель модулятора радиосигнала



Модель демодулятора радиосигнала



Исходные данные моделей

Время моделирования 1000.

Генератор случайных чисел:

Минимальное значение случайного числа =0; максимальное значение =1; начальное значение последовательности (любое целое число); Длительность одной реализации =100.

Minimum=0; Maximum=1; Initial seed=0; Sample time=100.

Генератор импульсов:

Период =100 секунд модели; Длительность импульса =50 (скважность равна 2); амплитуда 1 вольт; начальное время генерации =0.

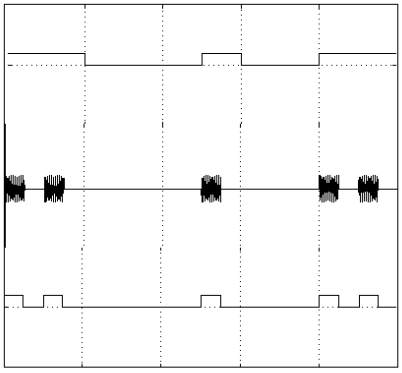
Period=100(secs); Duty cycle (% of period)=50; Amplitude=1; Start time=0.

Генератор синусоиды:

Амплитуда 1 вольт; Циклическая частота=10 радиан в секунду; Начальная фаза =0; Длительность одного значения 1.

Amplitude=1; Frequency (rad/sec)=10; Faze (rad)=0; Sample time=1.

Пример результатов моделирования



Практическое занятие №3
Построение элементов модели системы массового обслуживания

Цель занятия: Освоить методику моделирования элементов систем массового обслуживания (СМО)

1. Общие сведения о СМО

Системы массового обслуживания представляют собой класс математических схем, разработанных в теории массового обслуживания и различных приложениях для формализации процессов функционирования систем, которые по своей сути являются процессами обслуживания.

В качестве процесса обслуживания могут быть представлены различные по своей физической природе процессы функционирования экономических, производственных, технических и других систем, например потоки поставок продукции некоторому предприятию, потоки деталей и комплектующих изделий на сборочном конвейере цеха, заявки на обработку информации ЭВМ от удаленных терминалов и т. д. При этом характерным для работы таких объектов является случайное появление заявок (требований) на обслуживание и завершение обслуживания в случайные моменты времени, т. е. стохастический характер процесса их функционирования.

В любом элементарном акте обслуживания можно выделить две основные составляющие: ожидание обслуживания заявкой и собственно обслуживание заявки. Это можно изобразить в виде некоторого i-го прибора обслуживания $П_i$ (рис. 1), состоящего из накопителя заявок H_i в котором может одновременно находиться $l_i = \overline{1, L_i^H}$ заявок, где L_i^H — емкость *i*-го накопителя, и канала обслуживания заявок (или просто канала) K_i. На каждый элемент прибора обслуживания $П_i$ поступают потоки событий: в накопитель H_i — поток заявок w_i на канал K_i — поток обслуживании u_i.

Потоком событий называется последовательность событий, происходящих одно за другим в какие-то случайные моменты времени. Различают потоки однородных и неоднородных событий. Поток событий называется *однородным,* если он характеризуется только моментами поступления этих событий (вызывающими моментами) и задается последовательностью $\{t_n\} = \{0 \le t_1 \le t_2 \le \ldots \le t_n \le \ldots\}$, где t_n — момент

наступления n-го события — неотрицательное вещественное число. Однородный поток событий также может быть задан в виде последовательности промежутков времени между n-м и (n—1)-м событиями $\{\tau_n\}$, которая однозначно связана с последовательностью вызывающих моментов $\tau_n = t_n - t_n - t_{n-1},\ n \geqslant 1,\ t_0 = 0,$ т. е. $\tau_1 = t_1$.

Потоком неоднородных событий называется последовательность $\{t_n, f_n)$, где t_n — вызывающие моменты; f_n — набор признаков события. Например, применительно к процессу обслуживания для неоднородного потока заявок могут быть заданы принадлежность к тому или иному источнику заявок, наличие приоритета, возможность обслуживания тем или иным типов канала и т. п.

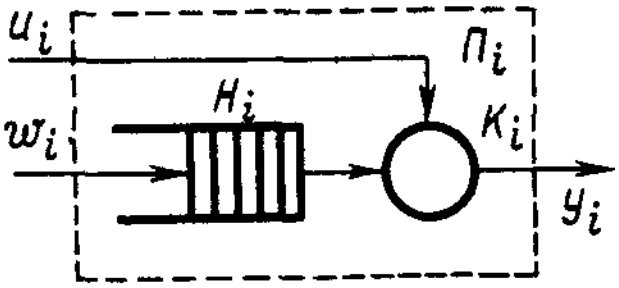


Рис. 1.Устройство обслуживания заявок

Рассмотрим поток, в котором события разделены интервалами времени $\tau_1, \tau_2, \ldots$, которые вообще являются случайными величинами. Пусть интервалы $\tau_1, \tau_2, \ldots$ независимы между собой. Тогда поток событий называется *потоком с ограниченным последействием*.

Пример потока событий приведен на рис. 2, где обозначено T_j интервал между событиями (случайная величина); T_n — время наблюдения, T_c —момент совершения события.

Интенсивность потока можно рассчитать экспериментально по формуле

$$\lambda = \frac{N}{T_n}$$

где N— число событий, произошедших за время наблюдения T_n . Если T_j —const или определено какой-либо формулой $T_j = f(T_{j-1})$, то поток называется *детерминированным*. Иначе поток называется *случайным*.

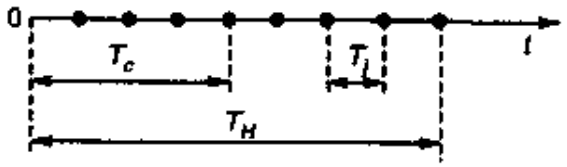


Рис.2 Графическое изображение *N-схемы*

Случайные потоки бывают:

— ординарными, когда вероятность одновременного появления 2-х и более событий равна нулю;

— стационарными, когда частота появления событий постоянная;

— без последействия, когда вероятность не зависит от момента предыдущих событий.

Поток событий называется *ординарным*, если вероятность того, что на интервал времени, примыкающий к моменту времени *t*, попадает больше одного события , пренебрежительно мала по сравнению с вероятностью того, что на этот же интервал времени попадает ровно одно событие.

Стационарным потоком событий называется поток, для которого вероятность появления того или иного числа событий на интервале времени *t* зависит от длины этого участка и не зависит от того, где на оси времени взят этот участок.

Интенсивность потока может быть любой неотрицательной функцией времени, имеющей размерность, обратную размерности времени. Для стационарного потока его интенсивность не зависит от времени и представляет собой постоянное значение, равное среднему числу событий, наступающих в единицу времени .

Возможные приложения. Обычно в приложениях при моделировании различных систем применительно к элементарному каналу обслуживания K_j, можно считать, что поток заявок $w_j \in W$, т. е. интервалы времени между моментами появления заявок (вызывающие моменты) на входе K_j образует подмножество неуправляемых переменных, а поток обслуживания $u_j \in U$, т. е. интервалы времени между началом и окончанием обслуживания заявки, образует подмножество управляемых переменных.

Заявки, обслуженные каналом K_j и заявки, покинувшие прибор $П_j$ по различным причинам необслуженными (например, из-за переполнения накопителя H_j, образуют выходной поток $y_j \in Y$, т. е. интервалы времени между моментами выхода заявок образуют подмножество выходных переменных.

Процесс функционирования прибора обслуживания $П_j$ можно представить как процесс изменения состояний его элементов во времени $z_j(t)$. Переход в новое состояние для $П_j$ означает изменение количества заявок, которые в нем находятся (в

канале K_j, и в накопителе H_j). Таким образом, вектор состояний для Π_j имеет вид $\vec{z}(t) = \left(z_j^K, z_j^H\right)$, где z_j^H — состояние накопителя (z_j^H = 0 — накопитель пуст, z_j^H =1 —в накопителе имеется одна заявка ... $z_j^H = L_j^H$ —накопитель полностью заполнен); L_j^H — емкость накопителя, измеряемая числом заявок, которые в нем могут поместиться; z_j^K — состояние канала (z_j^K =0 — канал свободен, z_j^K =l —канал занят и т. д.).

В практике моделирования систем, имеющих более сложные структурные связи и алгоритмы поведения, для формализации используются не отдельные приборы обслуживания, а *Q-схемы,* образуемые композицией многих элементарных приборов обслуживания (сети массового обслуживания). Если каналы *Ki* различных приборов обслуживания соединены параллельно, то имеет место многоканальное обслуживание (многоканальная *Q-схема),* а если приборы Π_j и их параллельные композиции соединены последовательно, то имеет место многофазное обслуживание (многофазная *Q-схема).* Таким образом, для задания *Q-схемы* необходимо использовать оператор сопряжения *R,* отражающий взаимосвязь элементов структуры (каналов и накопителей) между собой.

Связи между элементами *Q-схемы* изображают в виде стрелок (линий потока, отражающих направление движения заявок). Различают разомкнутые и замкнутые *Q-схемы.* В разомкнутой *Q-схеме* выходной поток обслуженных заявок не может снова поступить на какой-либо элемент, т. е. обратная связь отсутствует, а в замкнутых *Q-схемах* имеются обратные связи, по которым заявки двигаются в направлении, обратном движению вход-выход.

Собственными (внутренними) параметрами *Q-схемы* будут являться количество фаз обслуживания, количество каналов в каждой фазе , количество накопителей каждой фазы , емкость i-го накопителя . Следует отметить, что в теории массового обслуживания в зависимости от емкости накопителя применяют следующую терминологию для систем массового обслуживания: системы с потерями (т. е. накопитель отсутствует, а имеется только канал обслуживания *),* системы с ожиданием (т. е. накопитель, имеет бесконечную емкость и очередь заявок не ограничивается) и системы смешанного типа (с ограниченной емкостью накопителя. Всю совокупность собственных параметров *Q-схемы* обозначим как подмножество *H,*

Для задания *Q-схемы* также необходимо описать алгоритмы ее функционирования, которые определяют набор правил поведения заявок в системе в различных неоднозначных ситуациях. В зависимости от места возникновения таких ситуаций различают алгоритмы (дисциплины) ожидания заявок в накопителе и обслуживания заявок каналом , каждого элементарного обслуживающего прибора *П, Q-схемы.* Неоднородность заявок, отражающая процесс в той или иной реальной системе, учитывается с помощью введения классов приоритетов.

В зависимости от динамики приоритетов в *Q-схемах* различают статические и динамические приоритеты. Статические приоритета назначаются заранее и не зависят от состояний *Q-схемы,* т. е. они являются фиксированными в пределах решения конкретной задачи моделирования. Динамические приоритеты возникают при моделировании в зависимости от возникающих ситуаций. Исходя из правил выбора заявок из накопителя на обслуживание каналом, можно выделить относительные и абсолютные приоритеты.

Относительный приоритет означает, что заявка с более высоким приоритетом, поступившая в накопитель ожидает окончания обслуживания предшествующей заявки каналом *К*, и только после этого занимает канал. *Абсолютный приоритет* означает, что заявка с более высоким приоритетом, поступившая в накопитель прерывает обслуживание каналом *К* заявки с более низким приоритетом и сама занимает канал (при этом вытесненная из *К* заявка может либо покинуть систему, либо может быть снова записана на какое-то место в *Н*.

При рассмотрении алгоритмов функционирования приборов обслуживания (каналов, и накопителей) необходимо также задать набор правил, по которым заявки покидают *П* и *К*. Для *П*— либо правила переполнения, по которым заявки в зависимости от заполнения *Н* покидают систему, либо правила ухода, связанные с истечением времени ожидания заявки в *Н*, для *К*,— правила выбора маршрутов или направлений ухода. Кроме того, для заявок необходимо задать правила, по которым они остаются в канале *К* или не допускаются до обслуживания каналом *К* т. е. правила блокировок канала. При этом различают блокировки *К* по выходу и по входу. Такие блокировки отражают наличие управляющих связей в *Q-схеме,* регулирующих поток заявок в зависимости от состояний *Q-схемы*. Весь набор возможных алгоритмов поведения заявок в *Q-схеме* можно представить в виде некоторого оператора алгоритмов поведения заявок *А*.

Таким образом, *Q-схема,* описывающая процесс функционирования системы массового обслуживания любой сложности, однозначно задается в виде $Q = \langle W, U, H, Z, R, A \rangle$.

2. Моделирование основных элементов СМО с использованием пакета Simulink

В состав модели СМО в обязательном порядке входят модели генераторов потока заявок и потока обслуженных заявок. В основу моделей генераторов положены источники последовательностей случайных чисел равномерно распределенных на интервале (0…1). На основе этой последовательности можно имитировать потоки с различными законами распределения.

Пусть при моделировании некоторой системы необходимо сформировать на ЭВМ простейший поток заявок. Тогда длина интервала между *(i-1)*-м и *i*-м событиями $y_i = -(1/\lambda)\ln(x_i)$.

Если при моделировании некоторой системы требуется сформировать на ЭВМ поток событий, равномерно распределенных на интервале *(a,b),* то функция плотности интервалов между событиями *f(y)=1/(b—a), a ≤ y ≤ b.* Распределение первого интервала между началом отсчета и первым событием

$$f_1(y_1) = \lambda\left(1 - \int_0^{y_1} f(y)\,dy\right) = \lambda\left[1 - \int_0^{y_1} dy/(b-a)\right].$$

Интенсивность потока

$$\lambda = 1/M[y] = 1/\int_a^b y f(y)\,dy = 2/(a+b).$$

Тогда $f_1(y_1)=2[1-y_1/(b-a)]/(a+b)$.

где x_i — случайная величина, равномерно распределенная на интервале (0, 1).

Для формирования на ЭВМ потока Эрланга, в котором между последовательными событиями закон распределения интервалов

$$f_k(t)=\frac{\lambda(\lambda t)^{k-1}}{(k-1)!}e^{-\lambda t},\ t>0.$$

плотность распределения длины первого интервала

$$f_1(y_1)=a'(t_0,y_1)e^{-a(t_0,y_1)},$$

где a— математическое ожидание числа событий на интервале $(t_0+\Delta t)$, $a(t_0,y_1)=-ln(x_1)$.

Примерный состав модели генератора представлен на рис.2

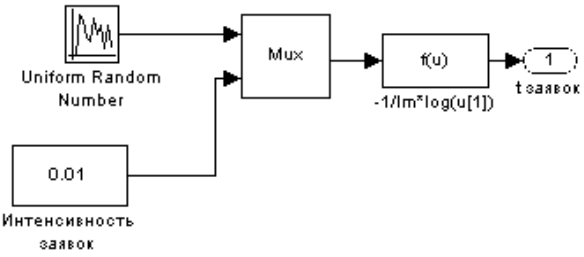


Рис.2 . Генератор

Функция, реализованная в генераторе, соответствует заданному закону распределения случайных величин. Генератор обслуженных заявок имеет аналогичную структуру.

Основной задачей накопителя является сравнение времени обслуживания заявки и времени поступления новой заявки. Если предыдущая заявка не была обслужена, формируется признак наличия задержанной заявки. На рис.3 показан примерный вид накопителя.

Вариант объединения накопителя и двух генераторов показан на рис.4.

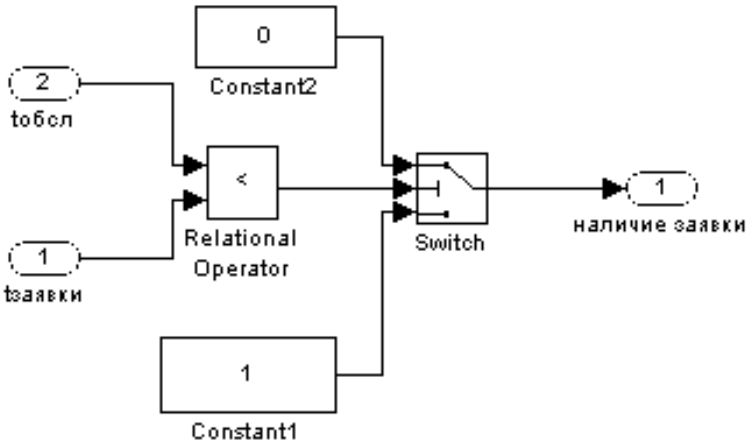


Рис. 3. Накопитель

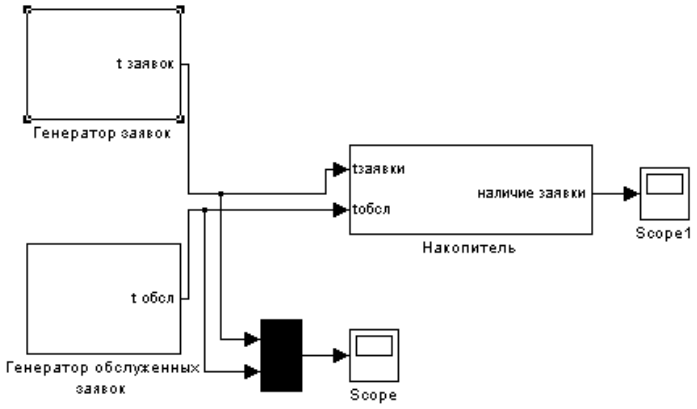


Рис. Пример соединения накопителя и генераторов

Практическое занятие №4.
Построение модели системы массового обслуживания

Цель занятия: Освоить методику моделирования элементов систем массового обслуживания (СМО)

В зависимости от характера источника заявок различают разомкнутые и замкнутые системы массового обслуживания СМО. На практическом занятии будут рассмотрены только разомкнутые СМО (рис.1). В зависимости от числа мест в очереди различают СМО с отказами и без отказов. В СМО с отказами число мест в очереди конечно и вследствие вероятностного характера, как входящего потока, так и процессов обслуживания, существует ненулевая вероятность того, что поступившая на вход СМО заявка застанет все каналы занятыми обслуживанием и все места в очереди занятыми

заявками, ожидающими обслуживания, т. е. она получит отказ. В СМО без отказов заявка либо сразу назначается на обслуживание, если в момент ее поступления свободен хотя бы один канал, либо безусловно принимается в очередь.

1. Построение сетевых моделей одноканальных систем массового обслуживания

Кратко рассмотрим методику составления сетевых моделей на примере одноканальной СМО (в совокупности с генератором заявок). Следует отметить, что процесс разработки сетевых моделей в общем случае является неформализованным. Проинтерпретируем СМО в терминах транзакций и ресурсов. Транзакции – это активные подвижные элементы системы, а ресурсы – неактивные. Транзакциями в СМО являются заявки, а ресурсом является канал обслуживания заявок. Функционирование СМО описывается как взаимодействие транзакций и ресурсов.

При переходе от системы транзакций и ресурсов к сетевой модели можно пользоваться следующими правилами. Каждый ресурс представляется позицией, причем маркировка этой позиции (простыми метками) определяет состояние ресурса. В одноканальной СМО имеется один ресурс, имеющий два состояния: "занят" и "свободен", причем начальное состояние ресурса – "свободен". Поставим этому ресурсу в соответствие позицию R сетевой модели. Маркировка $M(R) = 1$ будет свидетельствовать о незанятости ресурса, а $M(R) = 0$ – о его занятости. Маркировка $M(R) >= 2$ является запрещенной. Переход ресурса из одного состояния в другое представляется изменением маркировки позиции R: добавление метки соответствует освобождению ресурса, а изъятие – его занятию.

В системе транзакций и ресурсов каждая транзакция представляется простой меткой или меткой с атрибутами в зависимости от того, несет она информацию или нет. Будем считать, что в нашей одноканальной СМО каждая транзакция несет информацию и, следовательно, будет представляться меткой с атрибутами.

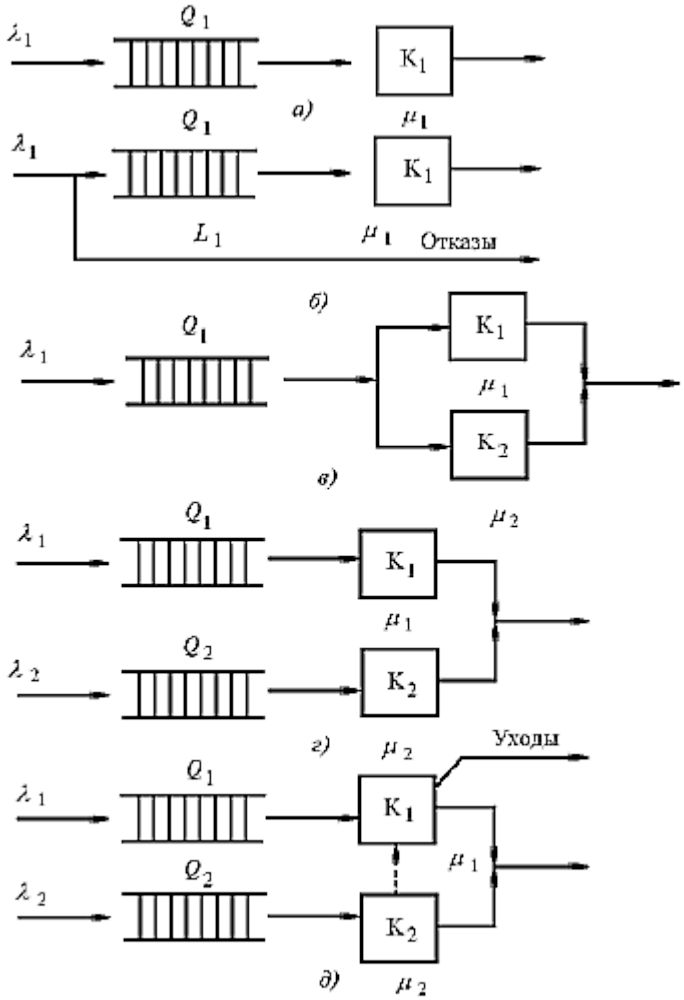


Рис. 1. Системы массового обслуживания:
а, *б* – одноканальные;
в, *г*, *д* – двухканальные

Транзакция в процессе ее обработки может находиться в нескольких состояниях, определяющих степень ее готовности. Для одноканальной СМО можно выделить следующие состояния транзакции:

0 – "готова для планирования";
1 – "планируется";
2 – "ожидание обслуживания";
3 – "заняла канал обслуживания";
4 – "обрабатывается";
5 – "обработана";
6 – "покинула канал обслуживания".

Состояния транзакции можно разделить на следующие классы:

– состояния временной активности (1, 4), связанные с процессами обработки транзакции, а также с планированием транзакции в генераторе транзакций. В этих состояниях транзакция находится во временной задержке перехода;

– состояния ожидания какого-либо ресурса (2). В этих состояниях транзакция находится в позициях, в которых нередко образуются очереди;

– состояния без ожидания (0, 3, 5, 6). В этих состояниях транзакция только "заявляет" о необходимости выполнения какого-либо события.

Каждому состоянию транзакции поставим в соответствие позицию или временную задержку перехода сетевой модели.

Переход транзакции из одного состояния в другое представляется как перемещение метки из одной позиции в другую, из позиции во временную задержку перехода, а также из временной задержки перехода в позицию.

В дальнейшем в моделируемой системе определяется множество событий, которые приводят к изменению состояния транзакций и ресурсов. В одноканальной СМО с генерацией заявок можно выделить следующие события:

1. "Приход новой заявки". При возникновении этого события транзакция переходит из состояния 1 в состояние 2. Создается копия транзакции, которая переводится в состояние 0.

2. "Планирование следующей заявки". Транзакция из состо-яния 0 переходит в состояние 1.

3. "Занятие канала обслуживания". Транзакция из состояния 2 переходит в состояние 3, а ресурс из состояния "свободен" переходит в состояние "занят".

4. "Начало обработки заявки". Транзакция из состояния 3 переводится в состояние 4.

5. "Конец обработки заявки". Транзакция из состояния 4 переводится в состояние 5.

6. "Освобождение канала обслуживания". Транзакция из состояния 5 переходит в состояние 6, а ресурс из состояния "занят" переходит в состояние "свободен".

7. "Уничтожение заявки". Транзакция покидает состояние 6 и уничтожается.

2. Моделирование очереди заявок

Очередью называется элемент исследуемого объекта, с помощью которого моделируются процессы ожидания начала обработки требований обслуживающими устройствами. Возникновение этих процессов происходит ввиду того, что в момент поступления требования устройство может находиться в состояниях "занято" либо "выключено". Кроме того, ожидание начала обработки может возникнуть из-за того, что приоритет поступившего требования меньше, чем приоритет обрабатываемого. По любой из вышеперечисленных причин, поступающие требования устанавливаются в очередь к устройству.

На входе устройства может формироваться произвольное количество очередей. При этом количество очередей не может превышать общего числа внесистемных и внутрисистемных источников, заданных в предложении ***входящий поток***. В зависимости от количества этих источников и правил функционирования элементов, имитирующих процессы ожидания, для описания очередей используется одна из следующих конструкций:

очередь: общая | отдельная | смешанная;

С помощью первой конструкции формируется одна очередь, в которую могут устанавливаться все требования, поступающие на вход устройства. Остальные две конструкции позволяют формировать множество очередей перед устройством. Независимо от количества очередей, установленных на входе устройства, дисциплины их формирования определяются предложением ***механизм обслуживания***.

Общая очередь

Общей очередью называется такая очередь, в которую могут устанавливаться требования, поступающие от всех внесистемных и внутрисистемных источников, указанных в предложении ***входящий поток***. Задание общей очереди осуществляется с помощью конструкции ***общая***. Формирование очереди производится независимо от способа передачи требований на вход устройства. Конкретное место, занимаемое в очереди каждым требованием, определяется в зависимости от его приоритета и механизма обслуживания. При этом основополагающим правилом является механизм обслуживания, устанавливающий дисциплину формирования очереди.

Пример модели обслуживающего устройства ***(n,1)*** с общей очередью представлен на рис. 1.

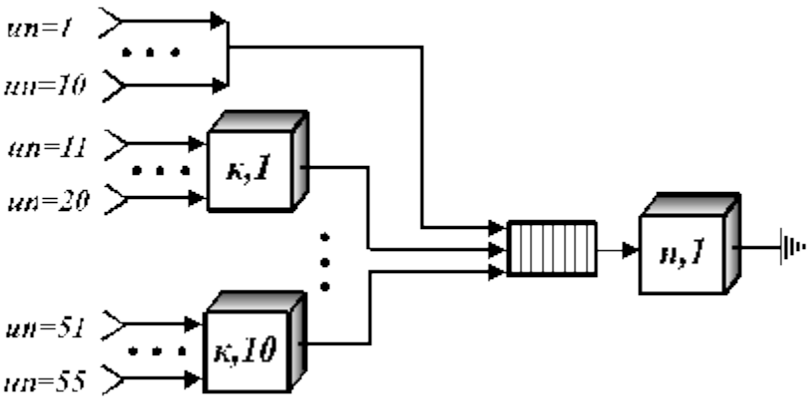


Рис. 1. Схема модели устройства с общей очередью

На вход устройства поступают требования от 10 внесистемных источников с индексами от 1 до 10 включительно и от 45 внутрисистемных источников. Интенсивность первых пяти источников требований равна 0.1, а следующих пяти источников - 0.2. Как показано на рисунке, на вход устройства ***(п,1)*** с устройства ***(к,1)*** поступают требования от внутрисистемных источников с индексами от 11 до 20 включительно, а с устройства ***(к,10)*** с индексами от 51 до 55 включительно. На устройствах ***(к,2)*** - ***(к,9)*** обрабатываются требования, генерируемые внесистемными источниками с индексами от 21 до 50 включительно. Описание обслуживающего устройства с общей очередью можно представить следующим образом:

у(п,1); вп: (ип=1-5 э(0.1)), (ип=6-10 э(0.2)), (ип=11-20 у(к,1)), ...(ип=51-55 у(к,10)); од: общая; мо: впп;

С применением средств интегрированного описания, позволяющих использовать фиктивные связи, модель устройства ***(п,1)*** можно представить в виде:

у(п,1); вп: (ип=1-5 э(0.1)), (ип=6-10 э(0.2)), (ип=11-55 у(к,1)2-10); од: общая; мо:впп;

Во втором случае при наличии одного внутрисистемного входа во входящем потоке конструкция ***од:общая;*** также позволяет сформировать общую очередь на входе ***(п,1)*** .

Отдельные очереди

Отдельными называются такие очереди, которые формируются автоматически для каждого внесистемного и внутрисистемного источника, заданного во входящем потоке.

При наличии нескольких очередей на входе устройства возникает вопрос об их приоритетах, который может решаться двумя способами: по правилу умолчания либо путем явного описания очередей в порядке убывания их приоритетов. Первый способ основывается на механизме обслуживания и порядке следования описаний внесистемных и внутрисистемных источников в предложении ***входящий поток***. При этом используется конструкция ***очередь: отдельная;***. Установка приоритетов очередям осуществляется следующим образом. Вначале формируются очереди для всех внесистемных источников. Приоритеты этих очередей определяются порядком следования описаний внесистемных источников во входящем потоке. Затем строятся очереди для внутрисистемных источников. Приоритеты этих очередей всегда ниже приоритетов очередей, формируемых внесистемными источниками. Установка приоритетов таким очередям также осуществляется в порядке очередности появления соответствующих описаний внутрисистемных источников во входящем потоке. В процессе моделирования приоритеты очередей, установленные по правилу умолчания, могут изменяться. Подробно процедуры модификации этих приоритетов освещены при описании механизма обслуживания. Задание приоритетов очередям с использованием правила умолчания рассмотрим на примере модели, представленной на рис. 2.

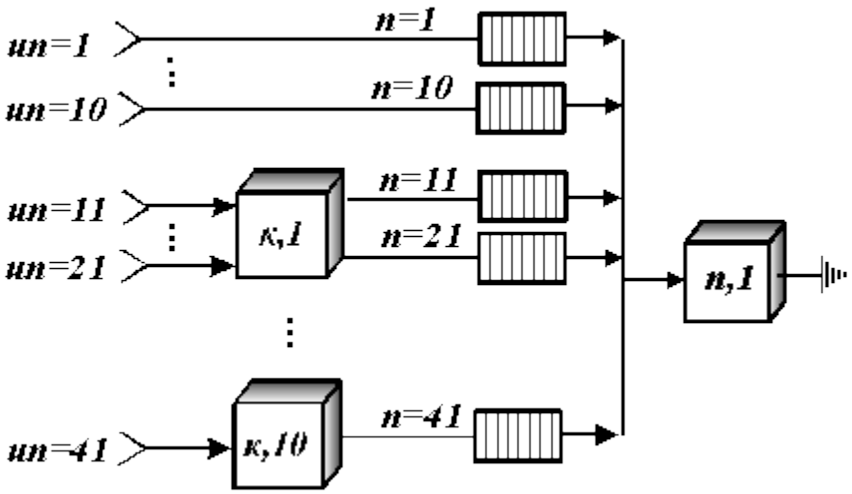


Рис. 2. Схема модели устройства с отдельными очередями

Как показано на схеме модели, на входе устройства ***(n,1)*** формируются отдельные очереди, образуемые десятью внесистемными потоками с индексами от 1 до 10 включительно и тридцатью внутрисистемными потоками с индексами от 11 до 41 включительно, поступающими с устройств ***(к,1)*** - ***(к,10)***. Описание этого фрагмента модели, иллюстрирующего формирование отдельных очередей, можно представить следующим образом:

у(n,1); вп:(ип=1-5 э(0.1)), (ип=6-10 э(0.2)), (ип=11 у(к,1)), (ип=12 у(к,1)), ... (ип=21 у(к,1)), ... (ип=41 у(к,10));

од: отдельная;

Согласно приведенному описанию на входе устройства ***(n,1)*** формируются отдельные очереди. Каждый внутрисистемный источник описывается независимо, что позволяет сформировать сорок одну очередь.

Наиболее простой способ модификации умалчиваемых значений приоритетов отдельных очередей состоит в изменении последовательности описания источников требований во входящем потоке. Предположим, что в рассматриваемом примере очередь на входе ***(n,1)***, формируемая 41 источником требований ***(ип=41 у(к,1))***, должна быть наиболее приоритетной среди очередей, образующихся из требований, поступающих от внутрисистемных источников. Для реализации такой модели описание входящего потока должно быть выполнено следующим образом:

у(n,1); вп:(ип=1-5 э(0.1)), (ип=6-10 э(0.2)),(ип=41 у(к,10)),
(ип=11 у(к,1)),(ип=12 у(к,1)), ..., (ип=40 у(к,9));
од: отдельная;

Приоритеты очередей, формируемых внесистемными источниками, также определяются согласно очередности появления соответствующих описаний. Поэтому для того, чтобы очередь, формируемая десятым внесистемным источником была наиболее приоритетной среди остальных очередей, входящий поток необходимо описать следующим образом:

у(n,1); вп:(ип=10 э(0.2)), (ип=1-5 э(0.1)), (ип=6-9 э(0.2)), (ип=41 у(к,10)),...;

При использовании правила умолчания для задания приоритетов очередям применение средств интегрированного описания внесистемных и внутрисистемных источников имеет одну особенность. Она состоит в том, что независимо от степени интеграции описания внесистемного входа для каждого из перечисленных в нем источников строятся отдельные очереди, а для внутрисистемного входа всегда создается одна очередь.

В тех случаях, когда невозможно или нецелесообразно использовать неявный способ задания приоритетов очередям, можно применить второй способ установки приоритетов. Он предполагает задание приоритетов очередям путем последовательного описания соответствующих входов в предложении ***очередь***:

отдельная
((<вход 1>) [<атрибуты очереди>])
[,((<вход 2>) [<атрибуты очереди>])
[..., ((<вход N>)[<атрибуты очереди>])]]

Представленная синтаксическая конструкция позволяет первой очереди, задаваемой с помощью конструкции ***<вход 1>***, установить наивысший приоритет, а последней очереди, ***<вход N>*** - наименьший. Например, конструкция ***од: отдельная ((ип=4)), ((ип=11-15 у(к,1))), ((ип=8-10));*** устанавливает убывающие приоритеты очередям, формируемым различными типами источников в следующем порядке:

1 очередь - ***((ип=4))***;
2 очередь - ***((ип=11-15) у(к,1)))***;
3 очередь - ***((ип=8))***;
4 очередь - ***((ип=9))***;
5 очередь - ***((ип=10))***;

Из приведенного примера видно, что для каждого внесистемного источника всегда формируется отдельная очередь, хотя в тексте модели задание третьей, четвертой и пятой очередей осуществляется с помощью интегрированного входа ***((ип=8-10))***.

При построении отдельных очередей не обязательно описывать все источники требований, заданные во входящем потоке. В этом случае алгоритм построения очередей и установки им необходимых приоритетов будет основываться на использовании двух указанных выше способов. Наиболее приоритетными будут очереди, формируемые на основе их явного описания. Для неописанных источников очереди создаются по правилу умолчания. Предположим, что описание входящего потока и очереди для устройства ***(п,1)*** представлено следующим образом:

вп: ***(ип=11-21 у(к,1)), (ип=22-24 у(к,2)), (ип=25-26 у(к,3)),***
(ип=27-28 у(к,4)), (ип=29-30 у(к,5)), (ип=31-32 у(к,6)),
(ип=33-34 у(к,7)), (ип=35-36 у(к,8)), (ип=37-40 у(к,9)),
(ип=41 у(к,10)),(ип=1-10 э(0.01));
од: ***отдельная ((ип=25-26 у(к,3))), ((ип=1)), ((ип=11-21 у(к,1)));***

Согласно этому описанию наивысшим приоритетом будет обладать очередь, формируемая из требований, поступающих с устройства ***(к,3)***. Второй по порядку приоритет будет иметь очередь, образующаяся из требований потока с индексом 1. Следующим по порядку приоритетом будет обладать очередь, формируемая из требований потоков с индексами от 11 до 21 включительно. Приоритеты остальным очередям задаются по правилу умолчания.

Смешанные очереди

Конструкция ***смешанная*** предназначена для формирования очередей, в которые могут устанавливаться требования, поступающие от различных типов источников. С помощью этой конструкции пользователю предоставляется возможность задавать различные правила объединения потоков для формирования очереди. Вслед за конструкцией ***смешанная*** необходимо указать, из каких источников должна формироваться первая очередь, затем вторая и т.д. Задание правила формирования очереди производится путем перечисления входов, образующих эту очередь. Далее может следовать описание атрибутов очереди, определяющих некоторые особенности ее функционирования. Синтаксис смешанной очереди представляется следующим образом:

смешанная
((<вход>)[,(<вход>)[...,(<вход>) [<атрибуты очереди>]]])
[,((<вход>)[,(<вход>)[...,(<вход>) [<атрибуты очереди>]]])
[...,((<вход>)[,(<вход>)[...,(<вход>)]] [<атрибуты очереди>])]]

Каждая из очередей, формируемых с использованием конструкции ***смешанная***, может образовываться из внесистемных и внутрисистемных источников. Ниже приведен пример модели, в которой на входе устройства ***(п,1)*** имеются две очереди (рис. 3).

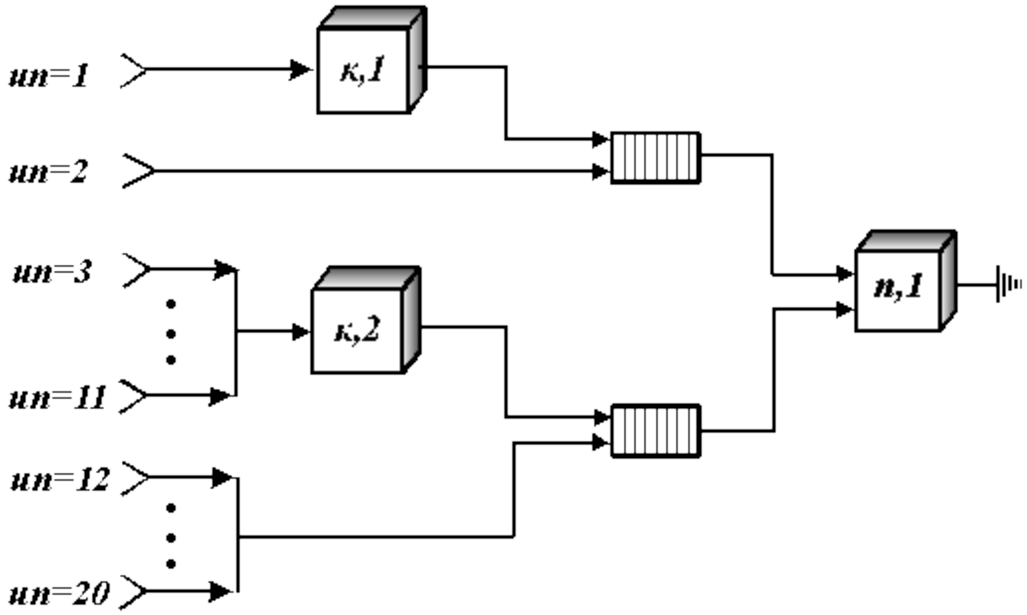


Рис. 3. Схема модели устройства со смешанными очередями

Первая из них формируется внутрисистемным потоком с индексом 1, поступающим с устройства ***(к,1)***, и внесистемным потоком с индексом 2. Вторая очередь также образуется из различных типов источников. При этом внутрисистемные потоки с индексами 3 - 11 поступают с устройства ***(к,2)***. Кроме того, в очередь могут устанавливаться требования, поступающие от девяти внесистемных источников с номерами 12 - 20. Описание фрагмента модели, иллюстрирующего задание смешанных типов очередей, представляется следующим образом:

у(п,1); вп: (ип=1 у(к,1)), (ип=2 э(0.1)), (ип=3-11 у(к,2)), (ип=12-20 э(0.2));

од: смешанная ((ип=1 у(к,1)), (ип=2)), ((ип=3-11 у(к,2)), (ип=12-20));

Если в описании смешанной очереди перечислены не все входы, то при построении очередей по умолчанию для каждого из них создается отдельная очередь. Предположим, что в рассмотренной выше модели для девятнадцатого и двадцатого потоков необходимо построить отдельные очереди. Для этого достаточно при описании правила формирования второй очереди опустить в перечислении девятнадцатый и двадцатый потоки:

у(п,1); вп: (ип=1 у(к,1)), (ип=2 э(0.1)), (ип=3-11 у(к,2)), (ип=12-20 э(0.2));

од: смешанная ((ип=1 у(к,1)), (ип=2)), ((ип=3-11 у(к,2)), (ип=12-18));

Этой же цели можно достигнуть путем прямого задания правила формирования третьей и четвертой очередей для 19-го и 20-го потоков:

у(п,1); вп: (ип=1 у(к,1)), (ип=2 э(0.1)), (ип=3-11 у(к,2)), (ип=12-20 э(0.2));

од: смешанная ((ип=1 у(к,1)), (ип=2)), ((ип=3-11 у(к,2)), (ип=12-18)), ((ип=19)), ((ип=20));

Если необходимо, чтобы 19 и 20-ый потоки образовывали одну очередь, то описание очереди должно быть представлено следующим образом:

од: смешанная ((ип=1 у(к,1)), (ип=2)), ((ип=3-11 у(к,2)), (ип=12-18)), ((ип=19), (ип=20));

Как и в случае с отдельным типом очереди, при наличии более одной очереди на входе устройства, возникает вопрос об их приоритетах. Приоритет задается согласно очередности описания правила формирования очереди. Первая очередь, встретившаяся в описании, имеет наивысший приоритет. Приоритет остальных очередей последовательно убывает. Если при описании правил формирования смешанной очереди указаны не все входы, перечисленные во входящем потоке, то для них по умолчанию также создаются отдельные очереди. При этом сначала строятся очереди для внесистемных, а затем - для внутрисистемных потоков.

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

	Авторы,	Заглавие	Издательство, год	Адрес	
Л1.1	Блинков, Ю. В.	Основы теории информационных процессов и систем: учебное пособие	Пенза: Пензенский государственный университет архитектуры и строительства, ЭБС АСВ, 2011	http://www.iprbookshop.ru/23103.html	
Л1.2	Белов, П. С.	Математическое моделирование технологических процессов: учебное пособие (конспект лекций)	Егорьевск: Егорьевский технологический институт (филиал) Московского государственного технологического университета «СТАНКИН», 2016	http://www.iprbookshop.ru/43395.html	
6.1.2. Дополнительная литература					
	Авторы,	Заглавие	Издательство, год	Адрес	
Л2.1	Казиев В. М.	Введение в анализ, синтез и моделирование систем	Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016	http://www.iprbookshop.ru/52188.html	
Л2.2	Плохотников, К. Э.	Методы разработки математических моделей и вычислительный	Москва: СОЛОН-ПРЕСС, 2017	http://www.iprbookshop.ru/64926.html	
6.1.3. Методические разработки					
	Авторы,	Заглавие	Издательство, год	Адрес	
Л3.1	Татарникова, Т. М.	Моделирование систем: методические указания к выполнению лабораторных работ	Санкт-Петербург: Российский государственный гидрометеорологический университет, 2008	http://www.iprbookshop.ru/12503.html	
Л3.2	Шевцова, Ю. В.	Математические модели и методы исследования операций: сборник задач	Новосибирск: Сибирский государственный университет телекоммуникаций и информатики, 2009	http://www.iprbookshop.ru/54766.html	
Л3.3	Сёмина, В. В.	Моделирование систем: методические указания для проведения лабораторных работ по дисциплине «моделирование	Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2016	http://www.iprbookshop.ru/64869.html	
6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"					
Э1	Душин В.К. Теоретические основы информационных процессов и систем [Электронный ресурс]: учебник/ В.К.— Электрон. текстовые данные.— М.: Дашков и К, 2014.— 348 с.— Режим д http://www.iprbookshop.ru/24764.— ЭБС «IPRbooks», по паролю				
Э2	Шатрова Г.В. Методы исследования и моделирования информационных процессов и технологий [Электр ресурс]: учебное пособие/ Шатрова Г.В., Топчиев И.Н.— Электрон. текстовые данные.— Ставрополь: С Кавказский федеральный университет, 2016.— 180 с				

ЭЗ	Лубенец Ю.В. Экономико-математические методы и модели [Электронный ресурс]: учебное пособие/ Лубенец Электрон. текстовые данные.— Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2013.-	

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Системная инженерия» для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Системная инженерия» содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии, направленность (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины – освоение методов разработки математических моделей информационных процессов и методологии и технологии математического моделирования при исследовании, проектировании, эксплуатации информационных систем; формирование общекультурных и профессиональных компетенций магистра в соответствии с требованиями ФГОС по направлению Информационные системы и технологии; подготовка магистра к деятельности, требующей применение научно-практических знанийи умений в области анализи информационных процессов; развитие логического, алгоритмического мышления студентов, умения самостоятельно расширять свои знания в области математического представления информационных процессов.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

В результате освоения данной дисциплины формируется следующая компетенция у обучающегося:

ОПК-6.3: Применяет основные положения системной инженерии и методы их приложения в области получения, передачи, хранения, переработки и представления информации посредством информационных технологий;

ОПК-6.1: Анализирует процессы получения, передачи, хранения и представления информации на основе положений системной инженерии;

ОПК-4.3: Анализирует существующие противоречия в практике при применении новых методов исследования.

Изучив данный курс, студент должен:

Знать:

- Принципы анализа и построения систем управления;
- Основные понятия и подходы в системной инженерии;
- Основные методы анализа и проектирования сложных систем;
- Базовые методы и средства системной и программной инженерии;
- ГОСТы и международные стандарты в области ИТ;
- Модели ЖЦ программных систем;
- Технологии и методологии проектирования программных систем;
- Порядок планирования и реализации модели ЖЦ при создании систем. Уметь:
- Определять назначение и технические характеристики системы с учетом целей ее создания;
- Сопоставлять назначение и технические характеристики системы с составом и функциональными возможностями ее компонентов.

Владеть:

- Проектирования элементов системной ИТ-архитектуры с использованием современ-ных CASE-средств;
- Планирования ЖЦ сложной системы;
- Формирования набора моделей, необходимых для создания систем;
- Принятия решений при выборе компонентов, необходимых для создания системы.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

1.1 Методы и средства проектирования информационных систем и технологий

Информационная система — система, предназначенная для сбора, хранения, обработки, поиска, распространения, передачи и предоставления информации (ГОСТ 7.0—99 п. 3.1.30).

Информационные системы и технологии(ИСТ), как и любые другие системы имеют определенный жизненный цикл - непрерывный процесс, который начинается с момента принятия решения о необходимости создания ИСТ и заканчивается в момент ее полного изъятия из эксплуатации. Жизненный цикл является моделью создания и использования ИСТ [1], которая отражает различные состояния системы с момента возникновения в данном комплексе средств до момента его полного выхода из употребления.

Основным нормативным документом, регламентирующим состав процессов ЖЦ систем, является международный стандарт - ISO/IEC 15288:2008 Standard for Systems Engineering — System Life Cycle Processes(Системная инженерия - процессы жизненного цикла систем). Российским аналогом является ГОСТ Р ИСО/МЭК 15288-2008 «Информационная технология. Процессы жизненного цикла систем» [1].

Среди множества концепций проектирования информационных систем в настоящее время следует выделить разработку, управляемую моделями (Model Driven Development, MDD). MDD - это развивающаяся парадигма, решающая многочисленные проблемы композиции и интеграции крупномасштабных систем и опирающаяся при этом на имеющиеся достижения в области технологий разработки программного обеспечения(в частности, на компонентное промежуточное программное обеспечение). MDD позволяет перевести разработку программного обеспечения на более высокий уровень абстракции по сравнению с тем, который возможен при использовании алгоритмических языков. Для представления элементов системы и их связей в подходе MDD используются модели. Модели служат входными и выходными данными на всех стадиях разработки, вплоть до генерации законченной системы.

Популярным вариантом MDD является модельно-управляемая архитектура (Model-Driven Architecture, MDA), предложенная и развиваемая консорциумом Object Management Group (OMG). В подходе MDA системы представляются с использованием языка моделирования общего назначения Unified Modeling Language (UML) и его конкретных профилей.

Начинается разработка с создания независимой от платформы модели(PIM). Затем после выбора языка программирования, исходя из специфики разработки, осуществляется трансформация PIM - модели в одну или несколько моделей, определяемых платформой, в рамках которой они реализуются(PSM).

Для разработки информационных систем и технологий по данной концепции могут использоваться различные методы и подходы. Подход к созданию ИСТ определяется набором составляющих его этапов, их последовательностью и используемыми на каждом этапе моделями. Кроме последовательности этапов подходы проектирования отличаются объектами исследования и синтеза. В зависимости от способа декомпозиции системы для проектирования информационных систем используется два основных подхода: ***структурный*** и ***объектно-ориентированный***.

Сущность структурного подхода к разработке ИСТ заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Элементами декомпозиции являются модули, связь между которыми реализуется через передачу управления. Система представляется совокупностью взаимодействующих модулей или процедур.

В объектно-ориентированном подходе основным элементом декомпозиции является объект, который может быть ассоциирован с объектом реального мира. Объект содержит данные о своих свойствах и состояниях, процедуры для изменения данных и связан с событиями, которые приводят к изменению его свойств. Система представляет собой совокупность взаимодействующих объектов.

Основными средствами проектирования информационных систем и технологий являются:

-языковые средства;

-инструментальные средства.

Среди множества языковых средств в настоящее время широкое распространение получил унифицированный язык моделирования(UML). UML это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования в основном программных систем. UML не является языком программирования, но в средствах выполнения UML-моделей как интерпретируемого кода возможна кодогенерация. Формальная спецификация последней версии UML 2.0 опубликована в августе 2005 года. Семантика языка была значительно уточнена и

расширена для поддержки методологии разработки, управляемой моделями. Последняя версия UML 2.4 опубликована в августе 2011 года. UML 1.4.2 принят в качестве международного стандарта ISO/IEC 19501:2005.

Язык UML имеет специальные базовые элементы графической нотации, которые необходимы для формирования всех модельных представлений. В частности в UML 2.0 существует 13(UML 2.2 - 14) официальных диаграмм, каждая из которых отражает различные аспекты системы.

Для того чтобы UML 2.0 поддерживал MDD, нужна платформа, имеющая обширные средства для адаптации языка манипулирования моделями. На поддержку MDD претендуют множество коммерческих инструментов UML. Лучшие из них поддерживают, в известных пределах, определение и использование трансформации моделей UML, но они ограничивают пользователей конкретной платформой реализации.

К наиболее популярным инструментальным средствам UML-моделирования относятся Rational Rose (IBM), Together (Borland) и MS Office Visio. Среди указанных средств особо следует выделить Borland Together, встраиваемое сегодня в распространенное RAD средство Borland Developer Studio(BDS), включающее Delphi, C++ Builder и C# Builder. Применение BDS позволяет в одной среде осуществлять как моделирование процессов и систем, так и создания приложений для их реализации на языках программирования высокого уровня.

Таким образом, MDD-разработка, основанная на MDA, в основном занимается трансформацией моделей и генерацией кода. С ее помощью разработчики сначала создают модель объекта на унифицированном языке моделирования, а затем генерируют код из этой UML-модели, применяя инструмент генерации кода. Модели, используемые для анализа и проектирования объектов информационных систем в языке моделирования UML, представляются в виде диаграмм.

1.2 Диаграммы унифицированного языка моделирования

Применение UML 2.0 позволяет разделить проблему моделирования сложной системы на составные части с помощью четырех представлений:

-статическое структурное представление модели описывает структурные аспекты системы, например, с помощью диаграммы классов;

-представление взаимодействия используется для моделирования последовательностей действий и коммуникаций, описывающих кооперацию взаимодействующих экземпляров;

-представление деятельности используется для создания моделей, описывающих поток «деятельностей» в системе;

-представление в виде конечного автомата используется для описания поведения системы в терминах состояний и переходов между ними.

Эти представления не являются полностью ортогональными: концепции, используемые в одном из них, часто зависят от концепций, применяемых в другом. Так, классификаторы участников взаимодействия должны быть определены в статической структурной модели. Такие зависимости определяются в метамодели UML, и инструментальные средства могут их задействовать для определения согласованности информации во всех представлениях системы.

Графические обозначения отдельных элементов моделей будут представлены при создании диаграмм в дальнейшем. Рассмотрим краткую характеристику диаграмм UML 2.0.

Структурные диаграммы

Диаграмма классов(Class diagram) — статическая структурная диаграмма, описывающая структуру системы, она демонстрирует классы системы, их атрибуты, методы и зависимости между классами.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

-концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;

-точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;

-точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).

Диаграмма компонентов(Component diagram) — статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между

компонентами. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п. Диаграмма компонента показывает структурные отношения между компонентами будущей информационной системы. В UML 2.0 компоненты являются автономными инкапсулированными единицами (unites) внутри системы или подсистемы, которые обеспечивают один или несколько интерфейсов. Поэтому диаграмма компонента позволяет архитектору убедиться в том, что компоненты реализуют заданную функциональность системы.

Диаграмма композитной/составной структуры (Composite structure diagram) — статическая структурная диаграмма, демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса. Подвидом диаграмм композитной структуры являются *диаграммы кооперации* (Collaboration diagram, введены в UML 2.0), которые показывают роли и взаимодействие классов в рамках кооперации. Кооперации удобны при моделировании шаблонов проектирования. Диаграммы композитной структуры могут использоваться совместно с диаграммами классов.

Диаграмма развёртывания(Deployment diagram) — служит для моделирования работающих узлов(аппаратных средств) и артефактов, развёрнутых на них. В UML 2 на узлах разворачиваются артефакты (*artifact*), в то время как в UML 1 на узлах разворачивались компоненты. Между артефактом и логическим элементом (компонентом), который он реализует, устанавливается зависимость манифестации.

Диаграмма объектов (Object diagram) — демонстрирует полный или частичный снимок моделируемой системы в заданный момент времени. На диаграмме объектов отображаются экземпляры классов (объекты) системы с указанием текущих значений их атрибутов и связей между объектами.

Диаграмма пакетов(Package diagram) — структурная диаграмма, основным содержанием которой являются пакеты и отношения между ними. Диаграммы пакетов служат, в первую очередь, для организации элементов в группы по какому-либо признаку с целью упрощения структуры и организации работы с моделью системы.

Диаграммы поведения

Диаграмма деятельности(Activity diagram) — диаграмма, на которой показано разложение некоторой *деятельности* на её составные части. Под деятельностью (англ. *activity*) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных *действий* (англ. *action*), соединённых между собой потоками, которые идут от выходов одного узла ко входам другого. Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений. Аналогом диаграмм деятельности являются схемы алгоритмов по ГОСТ 19.701-90.

Диаграмма автомата (State Machine diagram, *диаграмма конечного автомата*, *диаграмма состояний*) — диаграмма, на которой представлен *конечный автомат* с простыми состояниями, переходами и композитными состояниями. Конечный автомат — спецификация последовательности состояний, через которые проходит объект или взаимодействие в ответ на события своей жизни, а также ответные действия объекта на эти события. Конечный автомат прикреплён к исходному элементу (классу, кооперации или методу) и служит для определения поведения его экземпляров.

Диаграмма вариантов использования(Use case diagram) —представляет собой отражение действующих лиц (актантов), которые взаимодействуют с системой, и реакцию программных объектов на их действия. Актантами могут быть как пользователи, так и внешние агенты, которым необходимо передать или получить от них информацию. Значок варианта использования отражает реакцию системы на внешнее воздействие и показывает, что должно быть сделано для актанта. Основная задача — представлять собой единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.

Диаграммы взаимодействия

Диаграмма последовательности (Sequence diagram) — диаграмма, на которой изображено упорядоченное во времени взаимодействие объектов. В частности, на ней изображаются участвующие во взаимодействии объекты и последовательность сообщений, которыми они обмениваются. Диаграмма последовательности показывает хронологическую последовательность сообщений между объектами во взаимодействии. Она состоит из нескольких участников, таких как агенты, системы или подсистемы, классы и компоненты, представленные линиями жизни (lifelines), а также сообщения, которыми они обмениваются при взаимодействии.

Диаграмма коммуникации (Communication diagram, в UML 1.x — *диаграмма кооперации*, *collaboration diagram*) — диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. В отличие от диаграммы последовательности, на диаграмме коммуникации явно указываются отношения между элементами (объектами), а время как отдельное измерение не используется (применяются порядковые номера вызовов). Диаграмма коммуникации показывает поток сообщений между объектами и то, как несколько объектов сотрудничают при выполнении общей задачи. Как и диаграмма последовательности (sequence diagram), диаграмма коммуникации тоже может моделировать динамическое поведение для варианта использования (use case). Однако диаграмма коммуникации больше нацелена на показ того, как происходит координация, чем на хронометрирование последовательности.

Диаграммы коммуникации и последовательности транзитивны, выражают взаимодействие, но показывают его различными способами и с достаточной степенью точности могут быть преобразованы одна в другую.

Примечание.

По причине того, что диаграммы коммуникации и последовательности являются разными взглядами на одни и те же процессы, Rational Rose позволяет создавать из диаграммы коммуникации диаграмму последовательности и наоборот, а также производит автоматическую синхронизацию этих диаграмм.

Диаграмма обзора взаимодействия (Interaction overview diagram) — разновидность диаграммы деятельности, включающая фрагменты диаграммы последовательности и конструкции потока управления. Этот тип диаграмм включает в себя диаграммы последовательностей действий и диаграммы сотрудничества. Эти диаграммы позволяют с разных точек зрения рассмотреть взаимодействие объектов в создаваемой системе.

Диаграмма синхронизации(Timing diagram) — альтернативное представление диаграммы последовательности, явным образом показывающее изменения состояния на линии жизни с заданной шкалой времени. Эта диаграмма может быть полезна в приложениях реального времени.

Таким образом, выбранный разработчиком набор диаграмм позволяет создать практически любое представление о проектируемой системе.

Примечание.

Изображая диаграмму, воспользуйтесь следующими рекомендациями:

-дайте диаграмме имя, соответствующее ее назначению;

-расположите элементы так, чтобы свести к минимуму число пересечений;

-пространственно элементы расположите так, чтобы семантически близкие сущности располагались на диаграмме рядом;

-используйте примечания и цвет, чтобы привлечь внимание читателя к важным особенностям диаграммы.

Унифицированный процесс разработки программного обеспечения

Процесс проектирования ИСТ, кроме основных концепций и понятий, используемых при проектировании и реализации ИСТ, включает в себя технологию проектирования. Одной из развитых современных технологий является унифицированный процесс(Rational Unified Process,RUP). RUP – одна из лучших технологий разработки программного обеспечения, созданная в компании Rational Software, входящей в состав IBM. Унифицированный процесс позволяет создавать сложные программные системы, основываясь на индустриальных методах разработки [2, 3].

Вся разработка информационной системы (ИС) рассматривается в RUP как процесс создания артефактов. Любой результат работы проекта, будь то исходные тексты, объектные модули, документы, передаваемые пользователю, модели – это подклассы всех артефактов проекта.

Одним из интереснейших классов артефактов проекта являются модели, которые позволяют разработчикам определять, визуализировать, конструировать и документировать артефакты программных систем.

Модели позволяют рассмотреть будущую систему, ее объекты и их взаимодействие еще до вкладывания значительных средств в разработку, позволяют увидеть ее глазами будущих пользователей снаружи и разработчиков изнутри еще до создания первой строки исходного кода. Большинство моделей представляются диаграммами на унифицированном языке моделирования UML.

Основными моделями, создаваемыми в RUP, являются: модель вариантов использования, модель анализа, модель проектирования и модель реализации. Эти модели являются результатом основных работ процесса, к которым относятся: определение требований, анализ, проектирование и реализация. Рассмотрим содержание каждого из них.

1.3.1 Определение требований

Одним из важнейших этапов разработки ИС, согласно RUP, является этап определения требований, который заключается в сборе всех возможных пожеланий заказчика к работе системы. На данном этапе в ходе интервью с пользователями и изучения документов, аналитики должны собрать как можно больше требований к будущей системе, что не так просто, как кажется на первый взгляд. Позднее эти данные должны будут систематизированы и структурированы. Для того чтобы верно определить требования, разработчики должны понимать **контекст** (часть предметной области) в котором будет работать будущая система.

Определение контекста информационной системы

Для определения контекста ИСТ выполняется предпроектное обследование предметной области(области использования ИСТ). Для этого создаются модель предметной области и бизнес-модель, что является различными подходами к одному и тому же вопросу. Часто создается что-то одно: модель предметной области или бизнес-модель[2].

Отличия этих моделей в том, что модель предметной области описывает важные понятия, с которыми будет работать система и связи их между собой. Тогда как бизнес-модель описывает бизнес-процессы (существующие или будущие), которые должна автоматизировать(поддерживать) система. Поэтому кроме определения бизнес-объектов, вовлеченных в процесс, эта модель определяет работников, их обязанности и действия, которые они должны выполнять.

Использование UML не ограничивается моделированием программного обеспечения. Его также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Для создания модели предметной области с помощью UML используется обычная диаграмма классов, однако для создания бизнес-модели ее уже явно недостаточно. В этом случае применяется диаграмма вариантов использования с использованием дополнительных значков, которые отражают сущность бизнес-процессов – это бизнес-актант, бизнес-прецедент, бизнес-сущность и бизнес-управление. Эта модель намного ближе к следующей модели, создаваемой в процессе разработки – модели анализа.

При моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений часто используются диаграммы деятельности.

На практике диаграммы деятельности применяются в основном двумя способами:

-для моделирования процессов;

-для моделирования операций.

В первом случае внимание фокусируется на деятельности с точки зрения действующих лиц, которые работают с системой. Важным здесь является применимость диаграмм деятельности для описания бизнес-процессов. В данном случае для построения диаграмм деятельности используется так называемая траектория объекта, или поток объекта(object flow). Суть его состоит в том, что на диаграмме кроме деятельности можно изобразить и объекты, относящиеся к деятельности. С помощью символа зависимости(пунктирная стрелка) эти объекты можно соотнести с той деятельностью или переходом, где они создаются, изменяются или уничтожаются. Траектория объекта позволяет показать объекты, относящиеся к деятельности, а также моменты переходов этих объектов из одного состояния в другое.

Рекомендации по построению диаграмм деятельности для моделирования процессов заключаются в следующем.

Моделируют бизнес-процессы в несколько этапов, первым из которых является разбиение их на подпроцессы. Подпроцессы, являющиеся "участками большого процесса", описать легче.

Дальше выделяют ключевые объекты (и создают для них дорожки), определяют предусловия и постусловия каждого процесса (т. е. его границы), описывают деятельности и переходы, отображают на диаграммах состояния ключевых объектов, в которые они переходят в ходе процесса.

В итоге создается не какая-то абстрактная диаграмма, а модель реального бизнес-процесса в реальной компании, занимающейся реальным бизнесом.

Пример детализации конкретного бизнес-процесса с помощью диаграммы деятельности, созданной в Rational Rose, показан на рисунке 1. На диаграмме отражена деятельность выдачи товара со склада [1].

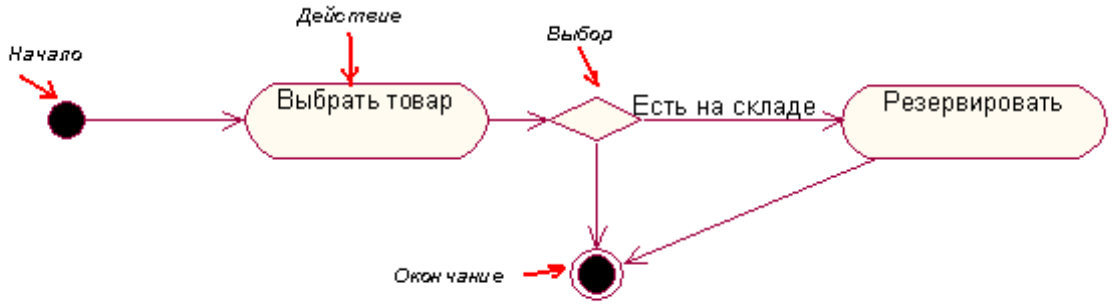


Рисунок 1 - Пример диаграммы активности, созданной в Rational Rose

Второй пример моделирования бизнес процесса оформление заказа в Интернет-магазине представлен на рисунке 2.

Рисунок 2 – Оформление заказа в Интернет-магазине

На рисунке 3 представлена диаграмма деятельности, выполненная в Borland Together. Здесь показана параметризованная деятельность с объектным узлом параметра, соединенным с контактами действий. Объектные узлы параметров размещаются на границе диаграммы, и ребра потока объектов соединяют их с контактами. Тип объекта, удерживаемого в объектном узле, обычно отображается в метке этого узла. В данном примере информация, используемая для заполнения заказа, предоставляется как входной параметр и передается в действие по вызову работы “Заполнение заказа”.

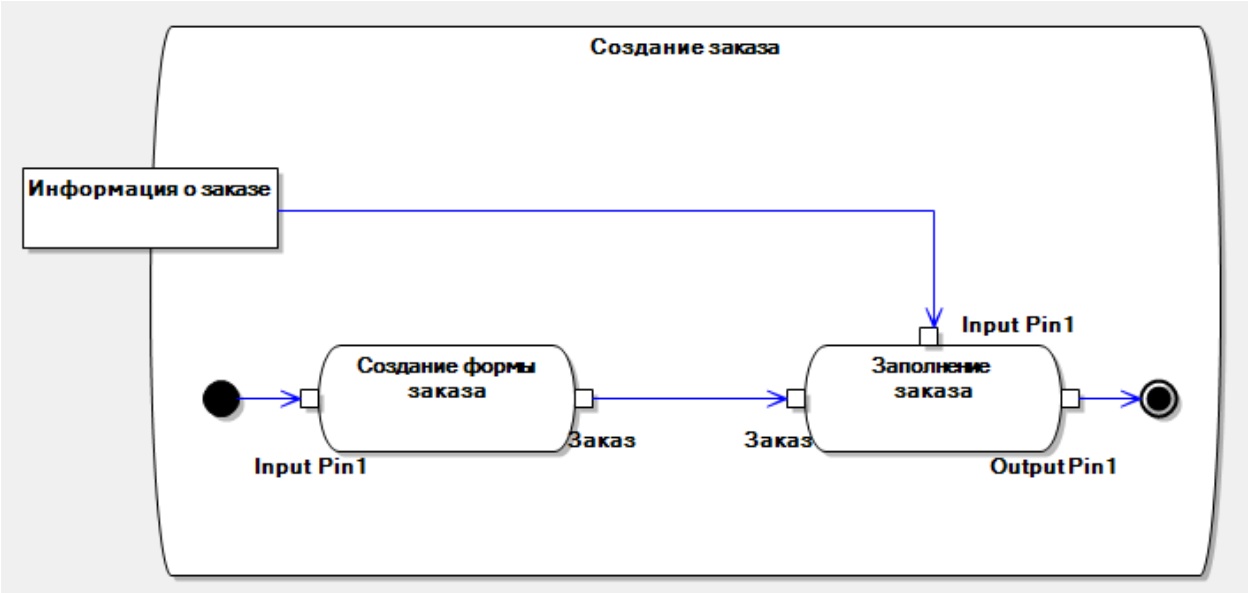


Рисунок 3 – Диаграмма активности, выполненная в Borland Together

Узлы управления в начале и в конце потока на рисунке 2 — это, соответственно, исходный и заключительный узлы. Когда вызывается деятельность “Создание заказа”, управляющий маркер помещается в начальный узел, а маркер данных с информацией о заказе — в объектный узел входного параметра. Управляющий маркер двигается от исходного узла к действию “Создание формы заказа”, которое начинает выполняться. Маркер данных передается от соответствующего параметра действию, вызывающему “Заполнение заказа”, которому приходится ждать до начала выполнения, пока “Создание формы заказа” не предоставит ему другие входные данные. После завершения действия “Заполнение заказа” управляющий маркер передается на конечный узел, деятельность завершается, а управление возвращается элементу, инициировавшему эту деятельность.

В случае моделирования операций диаграммы деятельности играют роль "продвинутых" блок-схем и применяются для подробного моделирования вычислений. На первое место при таком использовании выходят конструкции принятия решения, а также разделения и слияния потоков управления (синхронизации). Этот способ применяется при детализации вариантов использования и других процедур.

Рекомендации по построению диаграмм деятельности для моделирования операций заключаются в следующем.

Процесс построения диаграммы деятельности можно описать в виде последовательности таких действий:

1)Составление перечня деятельностей в системе

Как исходные данные для этой операции хорошо подходит список вариантов использования(или список операций). Дополняться диаграммой деятельности может каждый сценарий использования. Можно также попытаться описать связь между ними.

2)Определение зависимостей между деятельностями

Для каждой деятельности нужно найти деятельности, непосредственно предшествующие (и следующие за ней тоже), то есть деятельности, без выполнения которых поток управления не может перейти к данной деятельности.

3)Выделение параллельных потоков деятельностей

Выделяются деятельности, имеющие общих предшественников.

4)Определение условий переходов

Для этого формулируются выражения, которые могут принимать только два значения - "истинно" или "ложно", соответствующие альтернативным потокам управления.

5)Уточнение сложных деятельностей

Повторяя пункты 1-4 для каждой из деятельностей(при необходимости), можно уточнить сложные деятельности.

Таким образом, диаграммы деятельности в UML используются для моделирования потоков различного типа: потоков сигналов или данных, а также алгоритмических или процедурных потоков.

Пример диаграммы деятельности, выполненной в MS Office Visio, представлен на рисунке 4.

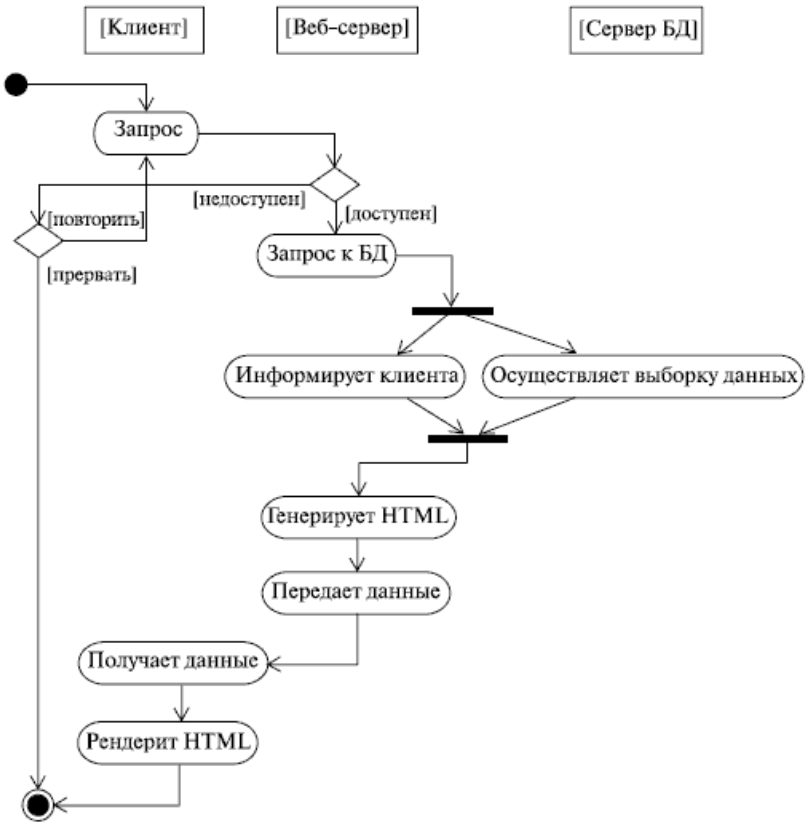


Рисунок 4 - Диаграмма деятельности, выполненная в MS Office Visio

На диаграмме показана работа с веб-приложением, решающим некую задачу в удаленной базе данных. Привлекает внимание расположение деятельностей на этой диаграмме: они как бы разбросаны по трем дорожкам, каждая из которых соответствует поведению одного из трех объектов - клиента, веб-

сервера и сервера баз данных. Благодаря этому легко определить, каким из объектов выполняется каждая из активностей, что очень упрощает ее восприятие.

Аналогия с дорожками действительно очень удачна. Именно таково официальное название элемента нотации UML, позволяющего указать распределение ролей на диаграмме деятельности.

Создавая диаграммы деятельности, необходимо учитывать, что они лишь моделируют срез некоторых динамических аспектов поведения системы. С помощью единственной диаграммы деятельности никогда не удастся охватить все динамические аспекты системы. Вместо этого следует использовать разные диаграммы деятельности для моделирования динамики рабочих процессов или отдельных операций.

Спецификация требований к информационной системе

Основным средством спецификации требований к проектируемой информационной системе в рамках RUP является модель вариантов использования. Главное назначение диаграммы вариантов использования заключается в формализации функциональных требований к системе. Основная задача — представить единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.

Модель варианта использования дает подробную информацию о поведении системы или приложения, которое разрабатывается. Она определяет требования к системе в терминах требующейся функциональности (вариантов использования) для достижения целей или для решения проблемы, определенной пользователем. Она же описывает окружение(агенты) и отношения между вариантами использования и агентами. Модель вариантов использования обычно включает в себя диаграммы вариантов использования и диаграммы действий, которые описывают то, как пользователи общаются с системой.

Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия внутренней структуры этой сущности. В качестве такой сущности может выступать исходная система или любой другой элемент модели, который обладает собственным поведением, подобно подсистеме или классу в модели системы.

Каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемую сущность или систему по запросу пользователя (актера), т. е. определяет способ применения этой сущности. Сервис, который инициализируется по запросу пользователя, представляет собой законченную последовательность действий. Это означает, что после того как система закончит обработку запроса пользователя, она должна возвратиться в исходное состояние, в котором готова к выполнению следующих запросов.

Примерами вариантов использования могут являться следующие действия: проверка состояния текущего счета клиента, оформление заказа на покупку товара, получение дополнительной информации о кредитоспособности клиента, отображение графической формы на экране монитора и другие действия.

Пример простейшей диаграммы вариантов использования “Заказ товара”, выполненной в Rational Rose, представлен на рисунке 5. На диаграмме показаны условные графические изображения главных элементов диаграммы – действующего лица(актанта) и варианта использования, а также связи между ними.

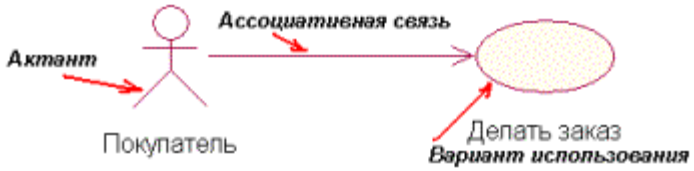


Рисунок 5 - Диаграмма вариантов использования,
выполненная в Rational Rose

Пример диаграммы вариантов использования, выполненной в Borland Together, показан на рисунке 6.

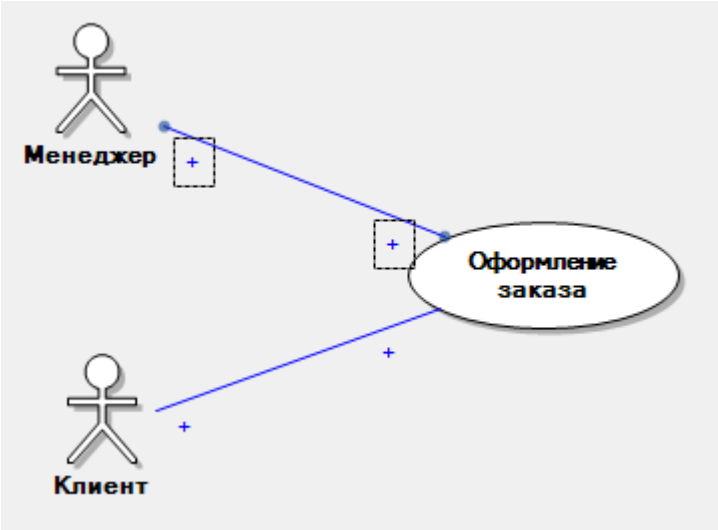


Рисунок 6 – Диаграмма вариантов использования, выполненная в Borland Together

Далее после создания диаграммы вариантов использования следует определить реализацию каждого варианта использования. Для этого применяются следующие способы:

-текстовое описание;

-описание алгоритма с помощью диаграмм деятельности;

-создание одной или несколько диаграмм взаимодействия.

Текстовое описание, соответствующее основной модели в рамках RUP, представляет собой:

-краткое описание;

-действующие лица;

-специальные требования;

-предпосылки;

-постусловия;

-точки расширения.

Например, рассмотрим вариант использования “Сделать предложение на аукционе”, диаграмма которого представлена на рисунке 7.

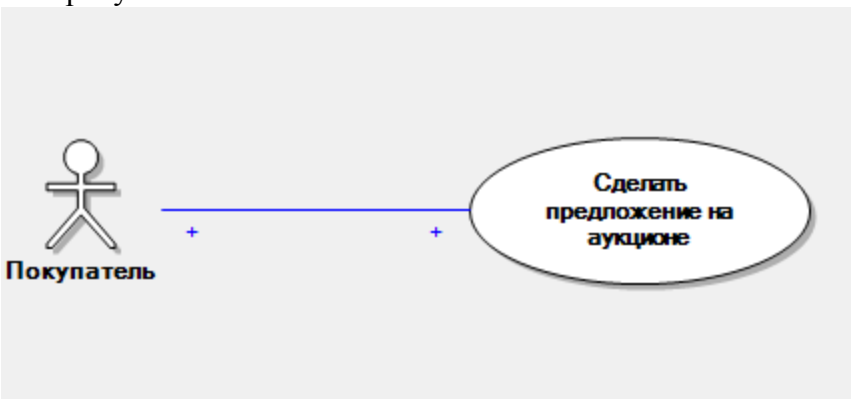


Рисунок 7 – Пример диаграммы варианта использования

В случае с приложением по ведению аукциона, речь может идти о представленном ниже потоке событий(последовательности, инициированной действующим лицом при подаче заявки в системе аукциона):

Основной поток:

-заявка (предложение цены): прецедент начинается в тот момент, когда покупатель предлагает свою цену на текущую позицию;

-ввод суммы: покупатель вводит сумму предложения. Система подтверждает, что сумма предложения превышает текущую ставку на значение, кратное шагу заявки для данной позиции;

-покупатель подтверждает заявку: покупатель подтверждает свое намерение разместить заявку;

-обработка заявки: система добавляет заявку к данной позиции;

-подтверждение заявки: система подтверждает наличие заявки путем отправки покупателю электронного сообщения. Продавец также уведомляется по электронной почте.

Альтернативные потоки операций могут описывать, что произойдет, если прием заявок будет прекращен до подачи предложения, если сумма заявки будет признана недействительной или покупатель не подтвердит подачу заявки.

Кроме текстового описания для детализации конкретного варианта использования(прецедента) можно построить диаграмму деятельности, правила построения которых аналогичны рассмотренных ранее.

Один из основных способов представления реализации варианта использования является создать одну или несколько диаграмм взаимодействия в форме диаграмм коммуникации или диаграмм последовательности, которые описывают один или несколько сценариев данного варианта использования. Этот способ в наибольшей степени соответствует идеологии UML и рекомендуется как основной и предпочтительный. Все эти операции выполняются на этапе анализа.

1.3.2 Анализ

После определения контекста, в котором будет работать система и требований к ней, наступает черед анализа полученных данных. В процессе анализа создается **аналитическая модель(модель анализа),** которая подводит разработчиков к архитектуре будущей системы. Аналитическая модель – это взгляд на систему изнутри, в отличие от модели вариантов использования, которая показывает, как система будет выглядеть снаружи.

Эта модель позволяет понять, как система должна быть спроектирована, какие в ней должны быть классы и как они должны взаимодействовать между собой. Основное ее назначение - определить направление реализации функциональности, выявленной на этапе сбора требований и сделать набросок архитектуры системы.

Модель анализа описывает логическую структуру системы и является фундаментом модели проектирования. Но в отличие от создаваемой в дальнейшем модели проектирования, модель анализа является в большей степени концептуальной моделью и только приближает разработчиков к классам реализации. Эта модель не должна иметь возможных противоречий, которые могут встретиться в модели вариантов использования.

Для построения аналитической модели выполняется анализ вариантов использования, который включает в себя:

-идентификацию классов, участвующих в реализации потоков событий;

-определение обязанностей классов;

-определение атрибутов и ассоциаций классов;

-унификацию классов анализа.

В потоках событий варианта использования выявляются классы трех типов:

-граничные классы, являющиеся посредниками при взаимодействии с внешними объектами;

-классы-сущности, представляющие собой основные абстракции (понятия) разрабатываемой системы;

-управляющие классы, обеспечивающие координацию поведения объектов в системе.

Классы анализа отражают функциональные требования к системе и моделируют объекты предметной области. Совокупность классов анализа представляет собой начальную концептуальную модель системы.

Для отображения модели анализа при помощи UML используется диаграмма классов со стереотипами (образцами поведения) «граничный класс», «сущность», «управление», а для детализации используются диаграммы взаимодействия, которые описывают взаимодействие групп объектов в различных условиях их поведения. Наиболее используемым типом таких диаграмм являются диаграммы коммуникации и последовательности.

Диаграмма коммуникации

Диаграмма коммуникации делает фокус на представлении группы взаимодействующих объектов и связей между ними, образующихся, если объекты общаются друг с другом посредством отсылки и приема сообщений. Также диаграммы коммуникаций подобны диаграммам объектов, но на них дополнительно могут быть показаны отсылаемые сообщения, причем допускается даже с указанием нумерации, описывающей порядок их следования во времени.

Диаграмма коммуникации используется для описания поведения системы как последовательности обмена сообщениями между элементами.

Основные сущности, используемые на диаграмме:

-роли, которые играют взаимодействующие элементы;

-объекты – экземпляры конкретных классов;

- связи - отношения, соединяющие взаимодействующие элементы.

Диаграмма коммуникации описывает поведение как взаимодействие, т. е. как протокол обмена сообщений между объектами.

Построение диаграммы коммуникации (кооперации) можно начинать сразу после построения диаграммы вариантов использования. В этом случае каждый из вариантов использования может быть специфицирован в виде отдельной диаграммы кооперации уровня спецификации.

Главная особенность диаграммы коммуникации (кооперации, сотрудничества) заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Прежде всего, на диаграмме коммуникации(кооперации, сотрудничества) в виде прямоугольников(окружностей) изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее должны быть изображены динамические связи - потоки сообщений. Они представляются в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

Пример диаграммы коммуникации (сотрудничества) показан на рисунке 8. Здесь показаны объект класса сущности “счет”, объект класса управления “обработчик” и объект граничного класса “интерфейс запроса на оплату”.

Стереотип «граничный класс» отображает класс, который взаимодействует с внешними актантами, «сущность» – отображает классы, которые являются хранилищами данных, а «управление» – классы, управляющие запросами к сущностям.

Линии, соединяющие объекты классов, отражают их взаимодействие.

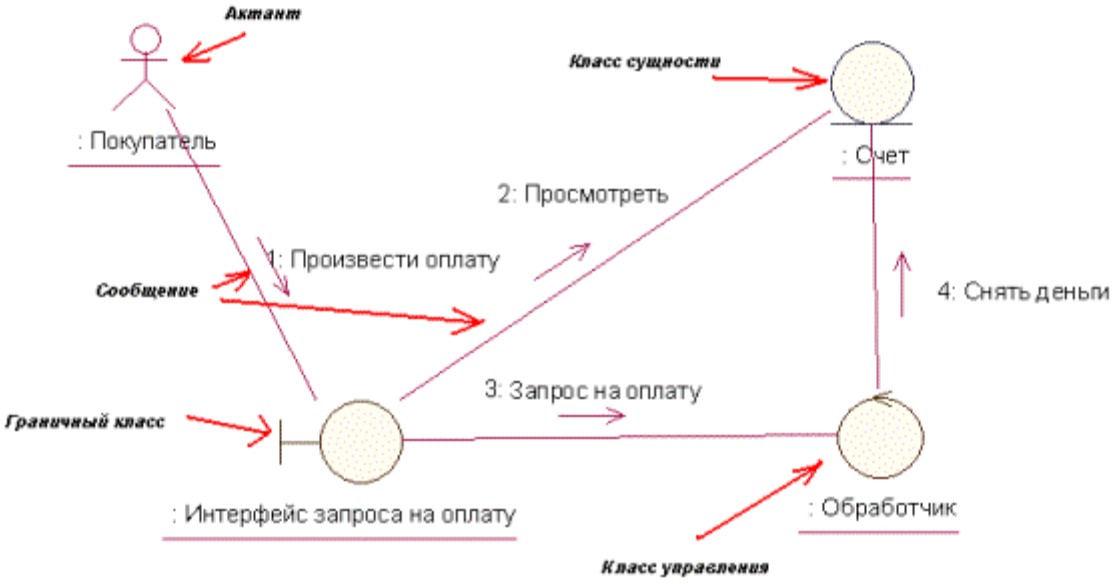


Рисунок 8 - Пример диаграммы коммуникации (сотрудничества)

Нумерация сообщений показывает их порядок, однако назначение диаграммы не в том, чтобы рассмотреть порядок обмена сообщениями, а в том, чтобы наглядно показать связи классов друг с другом.

При создании диаграммы коммуникации можно явно указать имена ассоциаций и ролей, которые играют объекты в данной ассоциации, как показано на диаграмме, изображенной на рисунке 9. Здесь показаны ассоциации “Продажа товара” и “Продажа компьютера”, а также роли “ клиент” и “менеджер”.

Рисунок 9 - Пример изображения ассоциаций и ролей

Диаграмма последовательности

Если акцентировать внимание на порядке взаимодействия, то другим его представлением будет диаграмма последовательности (Sequence).

Диаграмма последовательности - это диаграмма, чаще всего, описывающая один сценарий приложения. На диаграмме изображаются экземпляры объектов и сообщения, которыми они обмениваются в рамках одного варианта использования. Участники диаграммы именуются следующим образом: **имя: Класс**, где и имя, и класс являются не обязательными, но если используется класс, то присутствие двоеточия обязательно.

На диаграмме последовательности, каждый участник представлен вместе со своей линией жизни (lifeline), это вертикальная линия под объектом, вертикально упорядочивающая сообщения на странице. Важно: все сообщения на диаграмме следует читать сверху вниз. Каждая линия жизни имеет полосу активности (прямоугольники), которая показывает интервал активности каждого участника при взаимодействии.

Обозначение различных сообщений на диаграмме показано на рисунке 10.

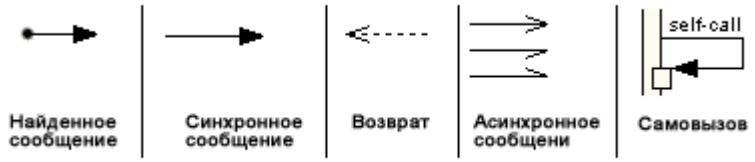


Рисунок 10 - Обозначение различных сообщений на диаграмме

У первого сообщения нет участника, пославшего его, поскольку оно приходит от неизвестного источника. Такое сообщение называется найденным сообщением (found message). Отправитель или получатель сообщения может находиться за пределами диаграммы коммуникации, и в этом случае используют входной и выходной шлюзы.

Сообщения, которыми обмениваются элементы, могут быть синхронными или асинхронными, что отражается в нотации стрелочек. Синхронные (synchronous message) - требующие возврата ответа, а асинхронные (asynchronous message) – ответа не требуют(вызывающий объект может продолжать работу). На диаграмме синхронные вызовы обозначаются закрашенными стрелочками, асинхронные – не закрашенными или половинными стрелочками.

Обратной пунктирной стрелкой показывается возврат ответа на сообщение(если сообщение является синхронным). Лучше применять изображение возврата только в тех случаях, когда это поможет лучше понять устройство взаимодействия. Во всех остальных случаях, стоит опускать изображения возвратов, т.к. они будут вносить некоторую неразбериху. Просто, при использовании синхронного сообщения, стоит помнить, что у него всегда есть возврат.

Если элемент диаграммы связан сам с собой, то такая связь называется рефлексивной(самовызов).

Если в сообщении требуется передать параметры, то они указываются в скобках через запятую, с указанием типа параметра (messageText(text : string)).

Для того чтобы задать порядок следования сообщений, используют десятичную нумерацию.

Пример диаграммы последовательности показан на рисунке 11. На ней представлены два объекта и все возможные виды сообщений.

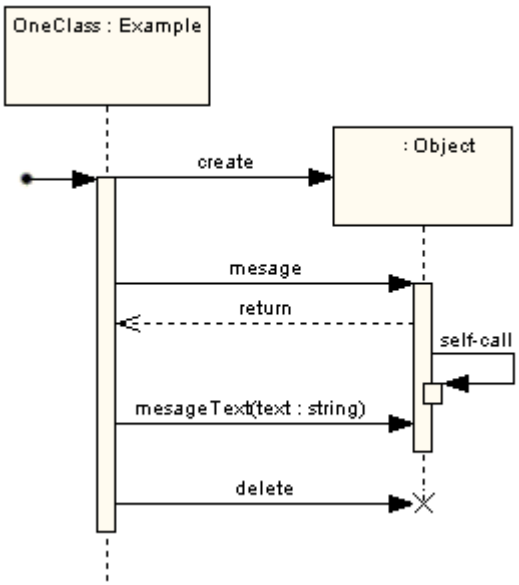


Рисунок 11 - Виды сообщений, которыми обмениваются объекты

Диаграмма последовательности, соответствующая диаграмме сотрудничества, показанной на рисунке 8, представлена на рисунке 12. Виды сообщений, которыми обмениваются объекты, аналогичны сообщениям диаграммы коммуникации, изображенным на рисунке 10.

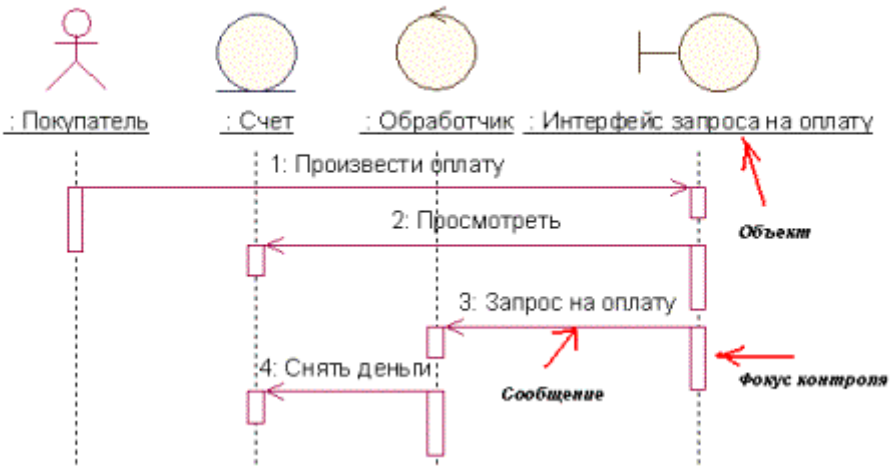


Рисунок 12 - Пример диаграммы последовательности действий

Решение о том какую из двух диаграмм нужно создавать первой, зависит от предпочтений конкретного разработчика. Поскольку эти диаграммы являются отображением одного и того же процесса, то и та и другая позволяют отразить взаимодействие между объектами.

Диаграммы коммуникации и последовательности транзитивны, выражают взаимодействие, но показывают его различными способами и с достаточной степенью точности могут быть преобразованы одна в другую.

При использовании такого инструмента для создания моделей как Rational Rose, эти два вида диаграмм могут быть созданы друг из друга автоматически [5].

В некоторых случаях могут строиться диаграммы обзора взаимодействия и диаграммы синхронизации.

1.3.3 Проектирование

Следующим этапом в процессе создания системы будет проектирование, в ходе которого на основании моделей, созданных ранее, создается **модель проектирования**. Эта модель отражает физическую реализации системы и описывает создаваемый продукт на уровне классов и компонентов. В отличие от модели анализа, модель проектирования имеет явно выраженную зависимость от условий реализации, применяемых языков программирования и компонентов.

Модель проектирования, используя различные диаграммы (в том числе диаграммы последовательности, машины состояний, компонента и размещения), подробно описывает, как устроено приложение и как оно будет реализовываться. Она также описывает структурные компоненты программ и технологий, например, обеспечивающих персистентность, распределение, безопасность и доступ к данным.

Для максимально точного понимания архитектуры системы, эта модель должна быть максимально формализована, и поддерживаться в актуальном состоянии на протяжении всего жизненного цикла разработки системы.

В RUP проектирование концентрируется вокруг определения архитектуры системы, а для систем с большой долей программного обеспечения - вокруг архитектуры программного обеспечения. Использование компонентных архитектур - один из шести наилучших подходов к разработке программ, инкорпорированных в RUP, рекомендует уделять больше времени на разработку и сопровождение архитектур. Время, затраченное на эти усилия, сокращает риски, связанные с ненадежными и негибкими системами.

Для создания модели проектирования используются целый набор UML диаграмм: диаграммы классов, диаграммы композитной структуры(кооперации), диаграммы взаимодействия, диаграммы активности. Основной является диаграмма классов.

Диаграмма классов

Диаграмма классов является основным типом диаграммы статической структуры. Она описывает структуру системы, показывая её классы, их атрибуты и операторы, и также взаимосвязи этих классов.

Каждый класс имеет имя, размещенное в верхнем блоке прямоугольника, изображающего класс. Для атрибутов и операций в элементах отводится отдельный блок. Каждый блок разделяется горизонтальной чертой.

Для атрибутов и операций применяются спецификаторы доступа. Спецификатора доступа языка C++ (public, private, protected) в UML отображаются символами + (public), - (private), # (protected), которые ставятся перед именем атрибута/операции. Также возможен вариант с ключевыми словами public, private, protected. Значение спецификаторов доступа: public - поля/методы класса видны снаружи класса. Т.е. к ним могут получать доступ объекты класса. private - поля/методы класса видны только внутри определения класса. protected - поля/методы класса видны в определении самого класса и в определениях производных классов.

Между классами существуют различные виды взаимодействия (или связи): один класс может быть производным другого, третий может содержать объект четвёртого в виде поля и т.д. Для различных видов взаимодействия в UML есть специальные названия.

Первый вид взаимодействия - ассоциация(association).

Ассоциация – это семейство связей двух и более классов. Обычно ассоциация возникает, когда один класс вызывает метод другого или если при вызове метода в качестве аргумента передаётся объект другого класса. Иногда при ассоциации показывают направленность (если это имеет значение).

Частным случаем ассоциации является связь – простая взаимосвязь между объектами. Она представляется линией соединяющей два или более объектных блока. Она встречается на диаграммах классов или объектов.

Всего существует пять типов ассоциации. Но наиболее распространены два: двунаправленная и однонаправленная ассоциации.

Сообщение направленная ассоциация(Message/Directed Association) используется, когда один класс “общается” с другим при создании экземпляра класса. Экземпляр класса — это описание конкретного объекта в памяти. Класс описывает свойства и методы, которые будет доступны у объекта, построенного по описанию, заложенному в класс. Экземпляры используют для представления конкретных сущностей реального мира.

Графически направленная ассоциация представляется в виде стрелочки направленной к “вызываемому” классу.

Частными вариантами ассоциации являются: агрегация и композиция.

Агрегация(быть частью) применяется, когда один класс должен быть контейнером других классов. Причем время существования содержащихся классов никак не зависит от времени существования класса контейнера. Графически агрегация представляется пустым ромбиком на блоке класса и линией, идущей от этого ромбика к содержащемуся классу.

Композиция - еще один случай ассоциации, но более строгий. В отличии от агрегации, композиция имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер будет уничтожен, то всё его содержимое будет уничтожено также. Графически представляется, как и агрегация, но с закрашенным ромбиком.

Различие между этими двумя видами ассоциации состоит в том, что композиция может быть частью одного и только одного целого, в то время как агрегация может быть частью нескольких объектов.

Еще одной разновидностью связи является **генерализация** (обобщение). Генерализация показывает, что один из двух связанных классов (*подтип*), является более частной формой другого (*супертип*), который называется обобщением первого. Графически генерализация представляется линией с пустым треугольником у супертипа.

Последнее отношение, которое мы рассмотрим, будет **реализация**(realization). Данная связь показывает отношение: класс - объект. На диаграмме реализация показывается пунктирной линией и не закрашенной стрелочкой.

Одной из важнейших характеристик взаимодействия является кратность(multiplicity) роли ассоциации. Кратностью роли ассоциации называется характеристика, указывающая, сколько объектов класса с данной ролью может или должно участвовать в каждом экземпляре ассоциации.

Наиболее распространенным способом задания кратности роли ассоциации является указание конкретного числа или диапазона. Например, указание “1” говорит о том, что все объекты класса с данной ролью должны участвовать в некотором экземпляре данной ассоциации, причем в каждом экземпляре ассоциации может участвовать ровно один объект класса с данной ролью. Указание диапазона “0..1” говорит о том, что не все объекты класса с данной ролью обязаны участвовать в каком-либо экземпляре данной ассоциации, но в каждом экземпляре ассоциации может участвовать только один объект. Аналогично, указание диапазона “1..*” говорит, что все объекты класса с данной ролью должны участвовать в некотором экземпляре данной ассоциации, и в каждом экземпляре ассоциации должен участвовать хотя бы один объект (верхняя граница не задана).

Пример диаграммы классов, на которой показаны все возможные варианты связей, представлен на рисунке 13.

Проектирование классов на данном этапе включает следующие действия:

-детализация проектных классов;

-уточнение операций и атрибутов;

-уточнение связей между классами;

-моделирование состояний для классов.

Детализация проектных классов, определенных в процессе анализа, уточнение атрибутов классов заключается в следующем:

-задается тип атрибута и значение по умолчанию (необязательно);

-задается видимость атрибутов: public, private или protected;

-при необходимости определяются производные (вычисляемые) атрибуты.

Обязанности классов, определенные в процессе анализа и документированные в виде «операций анализа», преобразуются в операции, которые будут реализованы в коде. При этом:

-каждой операции присваивается краткое имя, характеризующее ее результат;

-определяется полная сигнатура операции;

-создается краткое описание операции, включая смысл всех ее параметров;

-определяется видимость операции: public, private или protected;

-определяется область действия операции: операция объекта или операция класса.

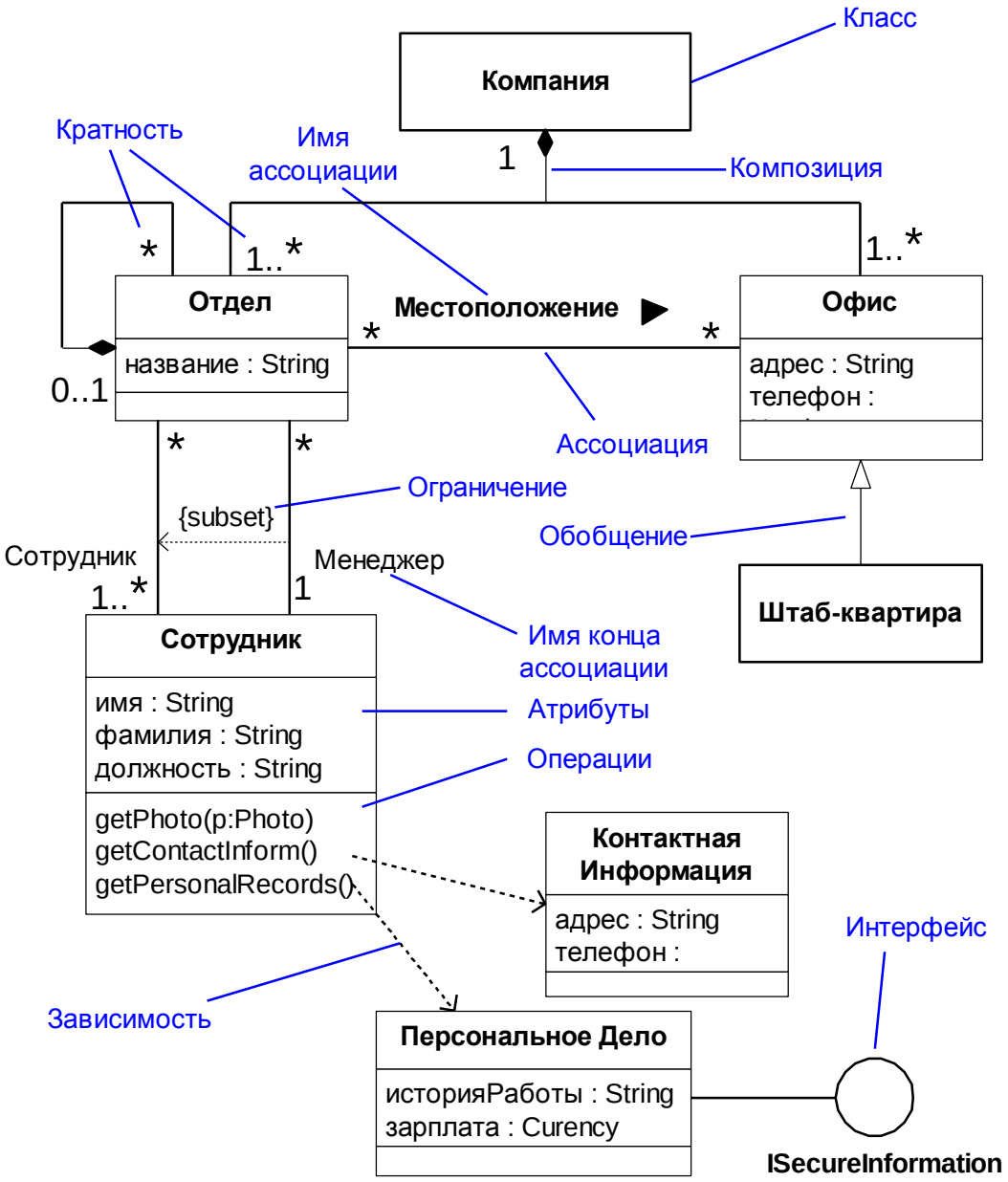


Рисунок 13 - Пример диаграммы классов

В процессе проектирования связи между классами подлежат уточнению. Ассоциации между граничными и управляющими классами преобразуются в зависимости. Агрегации, обладающие свойствами композиции, преобразуются в связи композиции. Связи обобщения могут преобразовываться в ситуациях с так называемой метаморфозой подтипов, когда объект суперкласса может менять свой подтип

Например, для бизнес-процесса – продажа товаров по заказу(каталогу), можно выделить классы Заказ и Клиент, который может представляться как юридическим, так и физическим лицом. Каждый класс имеет определенные атрибуты(свойства). Заказ выполняется в определенный день. Клиент имеет имя и т.д.

Пример диаграммы классов, созданной в среде Rational Rose, показан на рисунке 14.

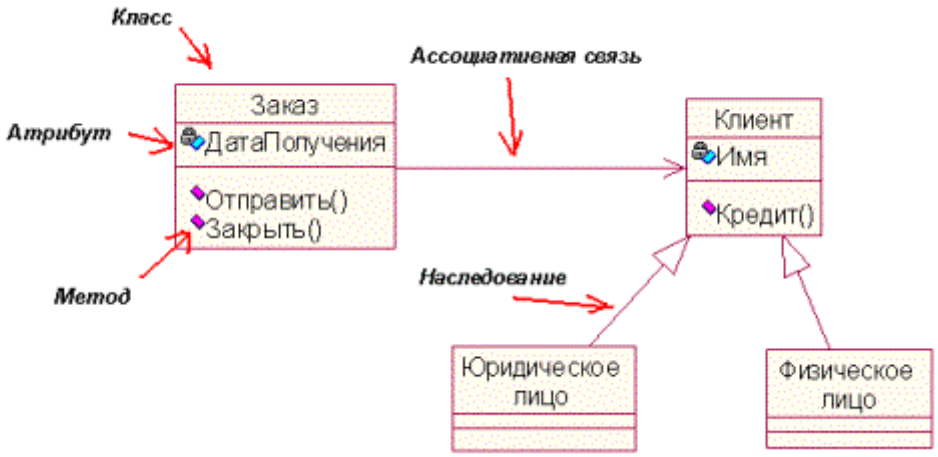


Рисунок 14 - Пример диаграммы классов, созданной в среде Rational Rose

Из диаграммы видно, что между классами “Заказ” и “Клиент” имеет место связь однонаправленная ассоциация.

Из диаграммы видно, что базовый класс “Клиент” имеет два производных класса “Юридическое лицо” и “Физическое лицо”, соединенных с базовым классом связью типа “Обобщение”, реализующей принцип наследования.

Другой вариант диаграммы классов для того же бизнес-процесса, созданной в среде Borland Together, показан на рисунке 15. На этой диаграмме классы имеют более полный набор атрибутов и операций. Однако класса Клиент не является базовым. Наследования классов на данной диаграмме не предусмотрено.

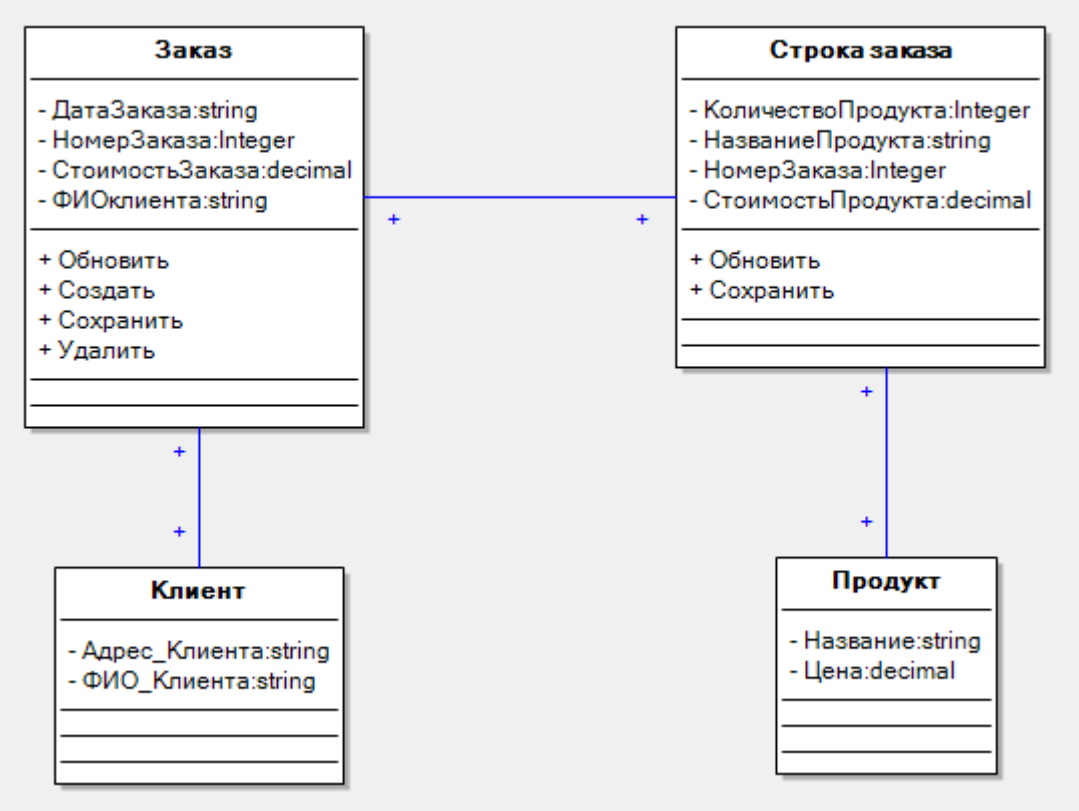


Рисунок 15 - Диаграмма классов, созданная в среде Borland Together

Основные правила построения диаграмм классов

В UML необязательно расписывать все детали классов. Это будет сделано при написании кода на конкретном языке (в нашем случае - C++). В UML-диаграмме можно опускать ненужные детали. Например, в диаграмму элемента можно добавить только те операции/атрибуты, которые важны для данной диаграммы, неважные особенности класса в UML можно опускать.

Для более полного раскрытия архитектуры проектируемой системы могут строиться диаграммы композитной/составной структуры и диаграммы автомата.

Диаграмма композитной/составной структуры

Диаграмма композитной/составной структуры — статическая структурная диаграмма, демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса.

Подвидом диаграмм композитной структуры являются диаграммы кооперации (Collaboration diagram, введены в UML 2.0), которые показывают роли и взаимодействие классов в рамках кооперации. Кооперации удобны при моделировании шаблонов проектирования.

Диаграммы композитной структуры могут использоваться совместно с диаграммами классов, как показано на рисунке 16.

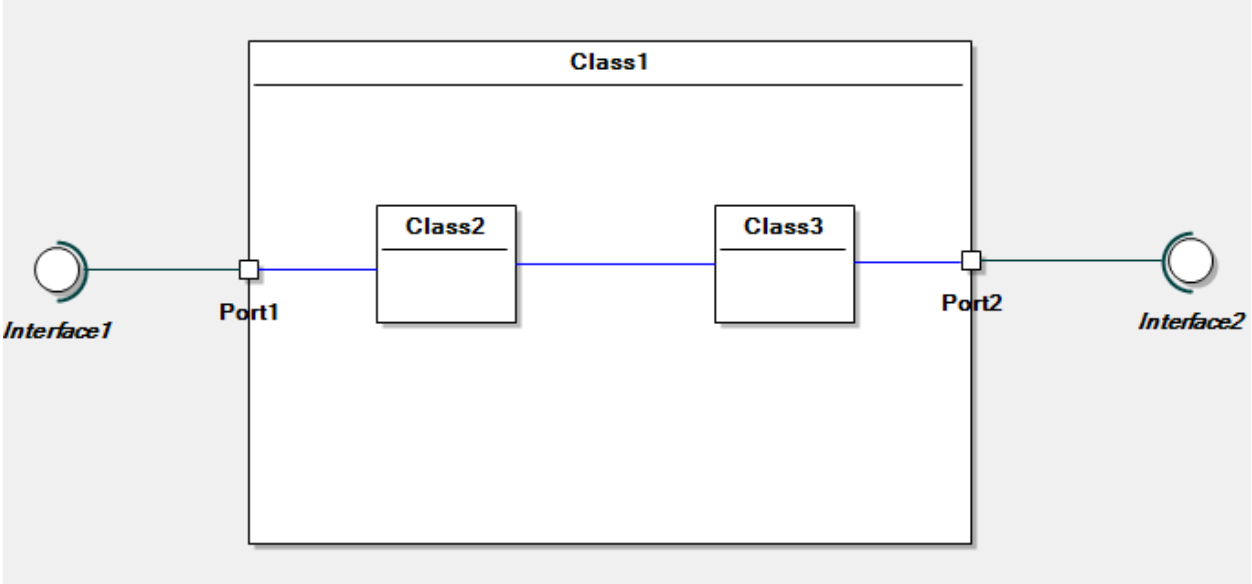


Рисунок 16 - Диаграмма композитной структуры, созданная в среде Borland Together

Диаграммы автомата

Диаграмма автомата, State Machine diagram (диаграмма конечного автомата, диаграмма состояний) — диаграмма, на которой представлен конечный автомат с простыми состояниями, переходами и композитными состояниями, как показано на рисунке 17.

Конечный автомат (State machine) — спецификация последовательности состояний, через которые проходит объект или взаимодействие в ответ на события своей жизни, а также ответные действия объекта на эти события. Конечный автомат прикреплён к исходному элементу (классу, кооперации или методу) и служит для определения поведения его экземпляров.

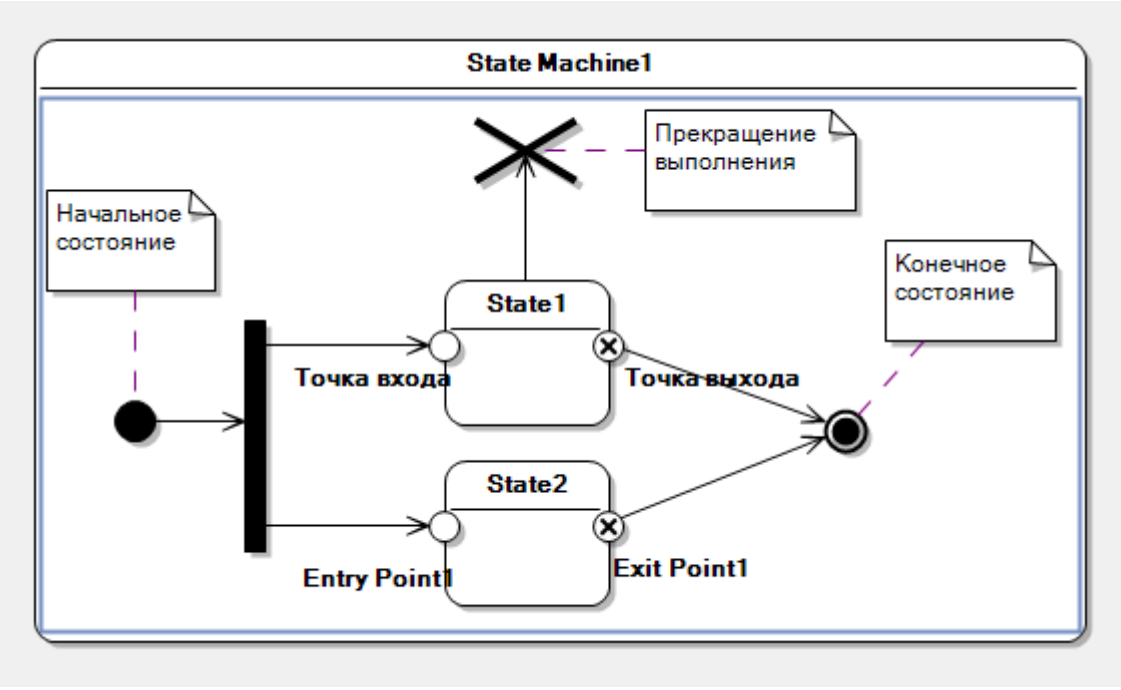


Рисунок 17 – Диаграмма автомата, созданная в среде Borland Together

Если в системе присутствуют объекты со сложным поведением, то строят диаграммы состояний. Построение диаграмм состояний может оказать следующее воздействие на описание классов:

-события могут отображаться в операции класса;

-особенности конкретных состояний могут повлиять на детали выполнения операций;

-описание состояний и переходов может помочь при определении атрибутов класса.

Каждая диаграмма состояний в UML описывает все возможные состояния одного экземпляра определенного класса и возможные последовательности его переходов из одного состояния в другое, то есть моделирует все изменения состояний объекта как его реакцию на внешние воздействия.

Диаграммы состояний чаще всего используются для описания поведения отдельных объектов, но также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

Главное предназначение этой диаграммы — описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

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

Действие рисуют вдоль линии перехода после имени события, его изображению предшествует наклонная (косая) черта.

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

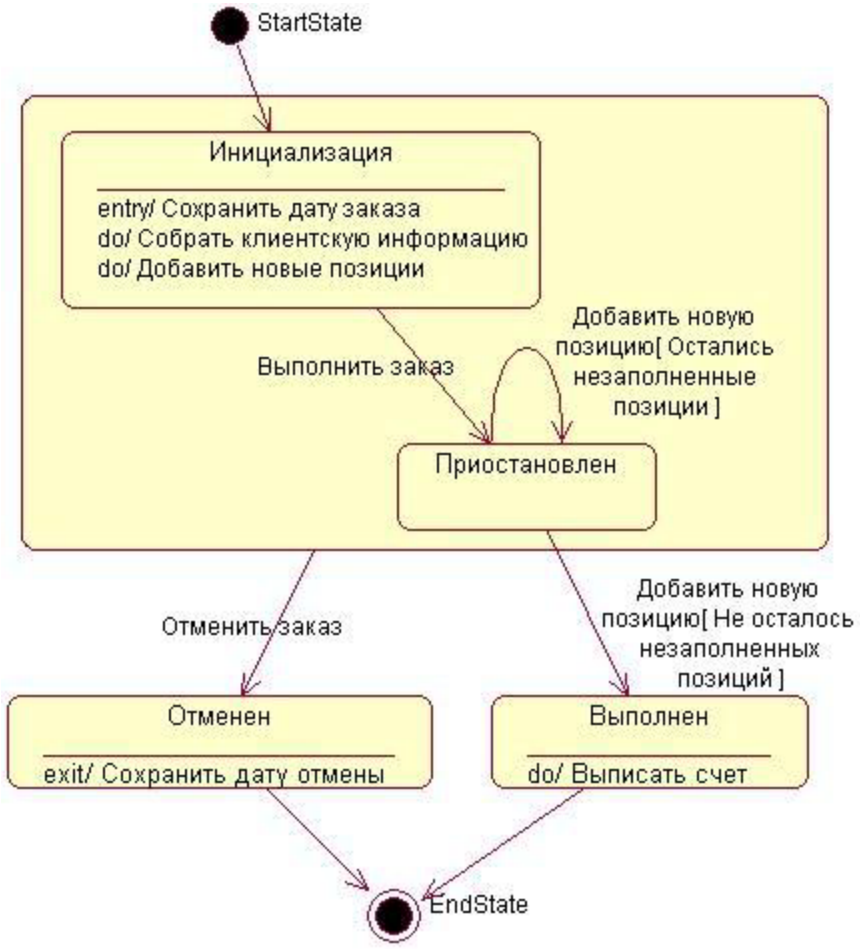


Рисунок 18 – Диаграмма состояний, созданная в среде Rational Rose

Для группировки классов, обладающих некоторой общностью, применяются пакеты. Пакет – общий механизм для организации элементов модели в группы. Каждый пакет – это группа элементов

модели, иногда сопровождаемая диаграммами, поясняющими структуру группы. Каждый элемент модели может входить только в один пакет. *Диаграммы пакетов* отображают зависимости между пакетами, возникающие, если элемент одного пакета зависит от элемента другого.

Пакеты также используются для представления подсистем. Подсистема – это комбинация пакета (поскольку она включает некоторое множество классов) и класса (поскольку она обладает поведением, т.е. реализует набор операций, которые определены в ее интерфейсах). Связь между подсистемой и интерфейсом называется связью реализации.

Жёсткого разделения между разными структурными диаграммами не проводится, поэтому данное название предлагается исключительно для удобства и не имеет семантического значения (пакеты и диаграммы пакетов могут присутствовать на других структурных диаграммах).

1.3.4 Реализация

Основная задача процесса реализации – создание системы в виде компонентов – исходных текстов программ, сценариев, двоичных файлов, исполняемых модулей и т.д. На этом этапе создается модель реализации, которая описывает то, как реализуются элементы модели проектирования, какие классы будут включены в конкретные компоненты. Данная модель описывает способ организации этих компонентов в соответствии с механизмами структурирования и разбиения на модули, принятыми в выбранной среде программирования и представляется диаграммой компонентов

Диаграмма компонентов

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Общий вид структуры информационной системы в виде диаграммы компонентов показан на рисунках 19 и 20. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ или реализации интерфейсов.

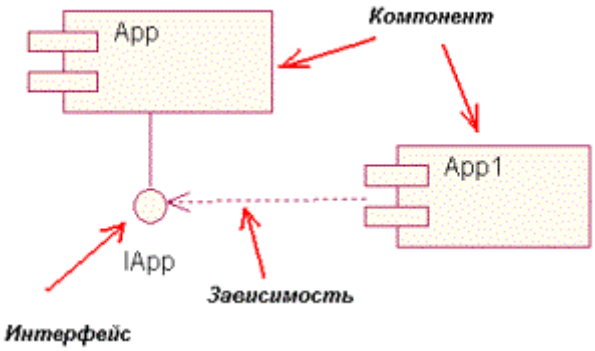


Рисунок 19 - Пример диаграммы компонентов на UML 1.5, созданной в среде Rational Rose

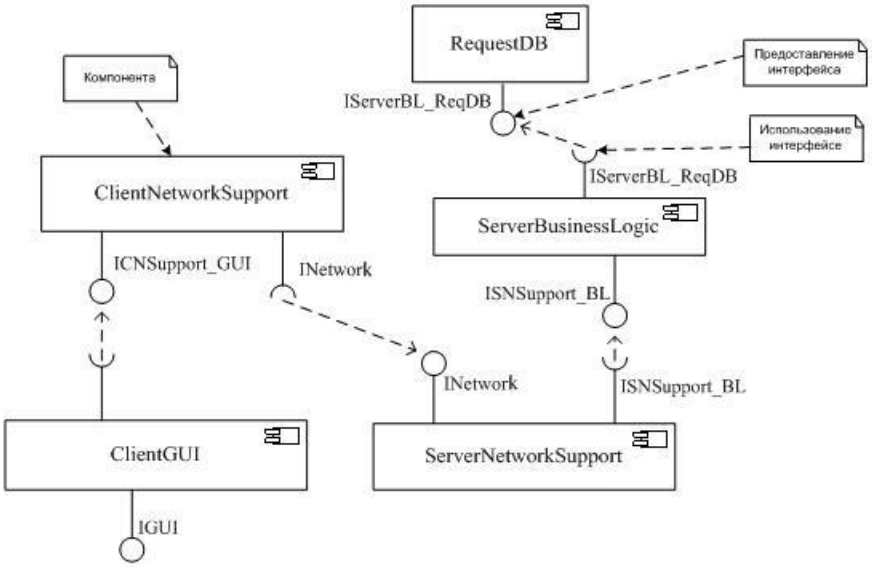


Рисунок 19 - Пример диаграммы компонентов на UML 2.0, созданной в среде Borland Together

Диаграмма развертывания

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Если разрабатывается программа, выполняющаяся локально на компьютере пользователя и не использующая периферийных устройств и ресурсов, то в разработке дополнительных диаграмм нет необходимости. При разработке же корпоративных приложений наличие таких диаграмм может быть крайне полезным для решения задач рационального размещения компонентов в целях эффективного использования распределенных вычислительных и коммуникационных ресурсов сети, обеспечения безопасности и других.

Для представления общей конфигурации и топологии распределенной программной системы в UML предназначены диаграммы развертывания.

Диаграмма развёртывания, Deployment diagram — служит для моделирования работающих узлов (аппаратных средств) и артефактов, развёрнутых на них. В UML 2 на узлах разворачиваются артефакты (artifact), в то время как в UML 1 на узлах разворачивались компоненты. Между артефактом и логическим элементом (компонентом), который он реализует, устанавливается зависимость манифестации. Это самый простой тип диаграмм, предназначенный для моделирования распределения устройств в сети. Для отображения используется всего два варианта значков процессор и устройство вместе со связями между ними.

Разработка диаграммы развертывания начинается с идентификации всех аппаратных, механических и других типов устройств, которые необходимы для выполнения системой всех своих функций. В первую очередь специфицируются вычислительные узлы системы, обладающие памятью и/или процессором.

Один из возможных вариантов построения диаграммы развертывания, созданной в среде Borland Together, показан на рисунке 21.

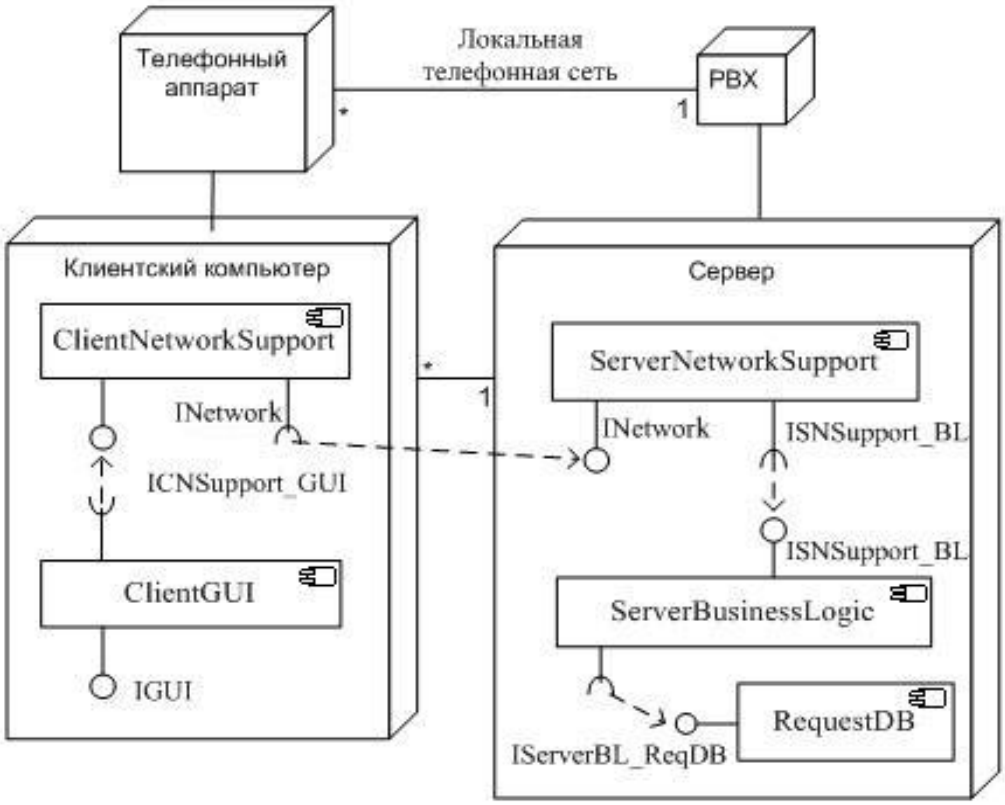


Рисунок 21 - Диаграмма развертывания, созданная в среде Borland Together

На диаграмме, показано каким образом компоненты телефонной службы приема заявок распределяются по аппаратной части системы.

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6.1. Рекомендуемая литература				
6.1.1. Основная литература				
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л1.1	Батоврин В. К.	Системная и программная инженерия. Словарь-справочник: Учебное пособие для вузов	Саратов: Профобразование, 2017	http://www.iprbookshop.ru/63956.html
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л1.2	Косяков А., Уильям Н., Сэмюэль Дж., Стивен М., Слинкин А. А.	Системная инженерия. Принципы и практика	Саратов: Профобразование, 2017	http://www.iprbookshop.ru/64063.html
6.1.2. Дополнительная литература				
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л2.1	Ехлаков, Ю. П.	Введение в программную инженерию: учебное пособие	Томск: Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2011	http://www.iprbookshop.ru/13923.html
Л2.2	Силич, В. А., Силич, М. П.	Теория систем и системный анализ: учебное пособие	Томск: Томский государственный университет систем управления и радиоэлектроники, 2011	http://www.iprbookshop.ru/13987.html

Л2.3	Кознов Д. В.	Введение в программную инженерию	Москва: Интернет-Университет Информационных Технологий	http://www.iprbookshop.ru/52146.html

6.1.3. Методические разработки

	Авторы, составители	Заглавие	Издательство, год	Адрес
Л3.1	Федоров, Ю. Н.	Справочник инженера по АСУТП. Проектирование и разработка: учебно-практическое пособие	Вологда: Инфра-Инженерия, 2016	http://www.iprbookshop.ru/5060.html
Л3.2	Данелян, Т. Я.	Теория систем и системный анализ: учебное пособие	Москва: Евразийский открытый институт, 2011	http://www.iprbookshop.ru/10867.html
Л3.3	Фролова, Е. А.	Методические указания по дисциплине Программная инженерия	Москва: Московский технический университет связи и информатики, 2013	http://www.iprbookshop.ru/61752.html

6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"

Э1	Павлов В.М. Искусство решать сложные задачи [Электронный ресурс]: системный подход/ Павлов В.М.— Электрон. текстовые данные.— М.: Дашков и К, 2015.— 184 c.— Режим доступа: http://www.iprbookshop.ru/35274.— ЭБС «IPRbooks»
Э2	Аверченков В.И. Мониторинг и системный анализ информации в сети Интернет [Электронный ресурс]: монография/ Аверченков В.И., Рощин С.М.— Электрон. текстовые данные.— Брянск: Брянский государственный технический университет, 2012.— 160 c.— Режим доступа: http://www.iprbookshop.ru/7001.— ЭБС «IPRbooks»
Э3	Букин Д.Н. Теория систем и системный анализ [Электронный ресурс]: учебное пособие/ Букин Д.Н.— Электрон. текстовые данные.— Волгоград: Волгоградский институт бизнеса, Вузовское образование, 2008.— 73 c.— Режим доступа: http://www.iprbookshop.ru/11351.— ЭБС «IPRbooks»

6.3.1 Перечень программного обеспечения

6.3.1.1	Windows 7 Корпоративная лицензионная по подписке Microsoft Imagine premium (оплата продления подписки Imagine premium по счету IM29470 от 28.01.2019г);
6.3.1.2	Kaspersky Endpoint Security 0E26-180226-121730-167-197;
6.3.1.3	Microsoft Office 2010 Professional Plus лицензионное соглашение № 49405992;
6.3.1.4	Консультант+ договор «Об информационной поддержке» № 1226/18 от 9.06.2018г. с сопровождением специалистами компании
6.3.1.5	Visual Studio 2013 лицензионное по подписке Microsoft Imagine premium оплата продления подписки Imagine premium по счету IM29470 от 28.01.2019г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Модели и методы поддержки принятия решений»
для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Модели и методы поддержки принятия решений» содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины - развить системное мышление у обучающихся путем детального анализа подходов к математическому моделированию и сравнительного анализа разных типов моделей. Ознакомить обучающихся с математическими свойствами методов и моделей оптимизации, которые могут использоваться при анализе и решении широкого спектра задач. Выработать у обучающихся навыки проведения численных исследований математических моделей и анализа результатов вычислений. Научить выбирать наиболее перспективное управляющее решение.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Изучив данный курс, студент должен:

Знать:

основы моделирования управленческих решений;

математические модели и информационные технологии процесса поддержки принятия решений;

многокритериальные методы поддержки принятия решений;

основные технологии информационной поддержки процесса поддержки принятия решений;

классификацию систем поддержки принятия решений и особенности используемых инструментальных средств;

современные методы и средства поддержки принятия решений в различных интеллектуальных системах,

принципы их рационального выбора в зависимости от особенностей процесса поддержки принятия решений.

Уметь:

осуществлять постановку конкретных задач поддержки принятия решений, выбирать адекватные математические и инструментальные средства их решения;

решать задачи, связанные с различными этапами подготовки и принятия решений в инструментальных системах

Владеть:

навыками формулирования требований к методам и моделям поддержки принятия решений;

навыками разработки отдельных их элементов;

навыками практического использования моделей и методов поддержки принятия решений;

навыками аналитического обоснования вариантов решений с использованием систем поддержки принятия решений.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическая работа 1. Исследование с помощью системы "Выбор"

Цель занятия:

1. Понятие СППР. Эволюция информационных технологий и информационных систем. Усвоить основные теоретико-информационные понятия учебной дисциплины изучить этапы развития информационной технологии и информационных систем.

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения

1. Дайте определение понятию конкурентные гонки?
2. Что такое информация?
3. Определите содержание СППР?
4. Чем отличаются данные от информации?
5. Основные характеристики данных?

Ход работы

1. Информация и данные.
2. Развитие информационных технологий.
3. Перспективные средства и направления развития информационных систем.
4. Основные понятия систем поддержки принятия решений.

Задача 1. Привести по три определения каждого понятия: “информация”, “данные”, “система поддержки принятия решений” (СППР).

Задача 2. Выделить критерии отбора альтернативных вариантов, которые, по вашему мнению, должны входить в состав СПР выбранной тематики. (например инвестиционные проекты: прибыль, срок окупаемости, и прочее). Также нужно выделить главные и второстепенные критерии. обосновать свой выбор. Тема определяется в соответствии с номером студента в академическом журнале (см темы разработки сппр).

Задача 3. Привести несколько существенных преимуществ применения СППР в выбранной области. Ответ обоснуйте.

Темы разработки сппр:

1. Создание проекта выбора ПК
2. Создание проекта выбора ТВ
3. Создание проекта выбора монитора
4. Создание проекта выбора микроволновой печи
5. Создание проекта выбора автомобиля
6. Создание проекта выбора магнитолы
7. Создание проекта выбора принтера
8. Создание проекта выбора сканера
9. Создание проекта выбора плоттера
10. Создание проекта выбора материнской платы
11. Создание проекта выбора процессора
12. Создание проекта выбора модема
13. Создание проекта выбора мобильного телефона
14. Создание проекта выбора программного обеспечения (по)
15. Создание проекта выбора винчестера (НЖМД)
16. Создание проекта выбора стационарного телефона
17. Создание проекта выбора DVD-проигрывателя
18. Создание проекта выбора интернет-провайдера
19. Создание проекта выбора видеокарты
20. Создание проекта выбора факсимильного аппарата

Практическая работа 2. Работа с СППР "Выбор"

Тема: работа с СППР ”Выбор". Основные функции, приемы и возможности

Цель занятия:

1. Ознакомиться с основными командами и получить базовые навыки при работе с СППР “Выбор”

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения

1. Назовите основные возможности СППР “Выбор”;

2. Перечислите основные команды, которые потребуются для создания нового работоспособного проекта.

3. Перечислите основные возможности окна разработки проектов.

4. Перечислите основные преимущества и недостатки СППР “Выбор”.

Ход работы:

1. Ознакомиться с основным принципами метода анализа иерархий.

2. Изучить основные аспекты интерфейса СППР “Выбор”.

3. Ознакомиться с интерфейсом и основными командами для пользования инструментами доступными в СППР “Выбор”.

4. Познакомиться с основными командами СППР “Выбор”.

5. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.

6. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

Основные аспекты интерфейса СППР "Выбор".

Система поддержки принятия решений (СППР) “Выбор”- аналитическая система, основанная на методе анализа иерархий (МАИ), является простым и удобным средством, которое поможет структурировать проблему, построить набор альтернатив, выделить факторы, характеризующие их, задать значимость этих факторов, оценить альтернативы по каждому из факторов, найти неточности и противоречия в суждениях лица принимающего решение (ЛПР эксперта), проранжировать альтернативы, провести анализ решения и обосновать полученные результаты. Система опирается на математически обоснованный метод анализа иерархий Томаса Саати.

Клиентское применения СППР “Выбор” обладает интуитивным пользовательским интерфейсом. Главное окно-это инструмент работы над проектом, позволяющий просматривать и редактировать выбранную иерархию проекта.

Основные компоненты интерфейса СППР " Выбор”

Панели инструментов.

С помощью панелей инструментов осуществляется быстрый доступ к основным инструментам приложения сосредоточенным в главном меню. Каждому пункту меню соответствует кнопка быстрого запуска на панели инструментов.

Подсказки.

Практически все элементы окон снабжены подсказками, которые появляются при небольшой задержке курсора мыши на необходимом элементе. Более подробная информация обо всех инструментах приложения содержится в помощи, сосредоточенной в главном меню Помощь.

Контекстно-зависимая помощь

СППР “Выбор” обладает мощной справочной системой по каждому инструменту приложения. Более 300 пунктов помощи помогут вам быстрее разобраться как использовать тот или иной элемент приложения. Помощь также предоставляет поиск информации по ключевым словам и фразам.

Помощь по диалоговым окнам

Все диалоговые окна снабжены справочной информацией. Нажмите клавишу F1 для вызова справки по текущему окну. Панели инструментов полностью копируют элементы главного меню. Каждому элементу главного меню соответствует панель инструментов:

* Файл-открытие, сохранение, закрытие проектов, создание новых, настройка приложения, принтера для печати, а также закрытие приложения.

• Проект - работа с иерархиями, произведение расчетов, получение отчетов, редактирование свойств проекта.

* Сеть-отсылка выбранным экспертам текстовых сообщений, проектов, открытие пришедших сообщений, просмотр списка сетевых событий.

* Помощь-вызов справки и окна информации о приложении.

В главном окне приложения для быстрого вызова некоторых инструментов можно использовать следующие горячие клавиши:

Файл

Ctrl + n-новый
F3-Открыть
F2-Сохранить
Ctrl + f2-Сохранить как
F10-Закрыть
F4-настройка принтера
Ctrl+o - Опции

Проект

Ctrl + i - информация о готовности
Ctrl + c-расчет
Ctrl + alt + c - сетевой расчет
Ctrl + r-результаты расчетов
Ctrl + p-просмотр отчета
Ctrl + t-изменить типа
Ctrl+v - Режим просмотра
Shift+Ctrl + P - Свойства

Сеть

F5-текстовое сообщение
F6-послать проект
F10-выбрать экспертов
F9-Эксперты
F11 - Окно сетевых сообщений
F12-окно с пакетами

Помощь

F1-Информация
Ctrl + f1-о программе

В окнах со списками, если фокус ввода находится на списке, то становятся доступными следующие горячие клавиши:

• Insert или "+" - добавление записи.

* Delete или " - " - удаление текущей записи.

Практическое задание.

Создать проект выбора автомобиля имея такую информацию:

Марки автомобилей 5-10 шт.

Критерии оценки:

1. Мощность двигателя (мощный)
2. Цена (Высокая)
3. Качество (Высокое)
4. Ведомость марки (Известная)
5. Комфортность (Комфортная)
6. Стильность (Стильная)

Создание проекта

1. Командой “Файл” – “Новый...” – “Простой проект” создать новый проект.

2. Щелкнуть на прямоугольнике по центру окна правой кнопкой мыши, вызвав при этом контекстное меню и в нем выбрать команду “Свойства проекта”.

3. Выбрать вкладку "Уровни".

4. Щелкнуть на знаке “+” 3 раза.

5. В списке появится 3 уровне (рис. 1.).

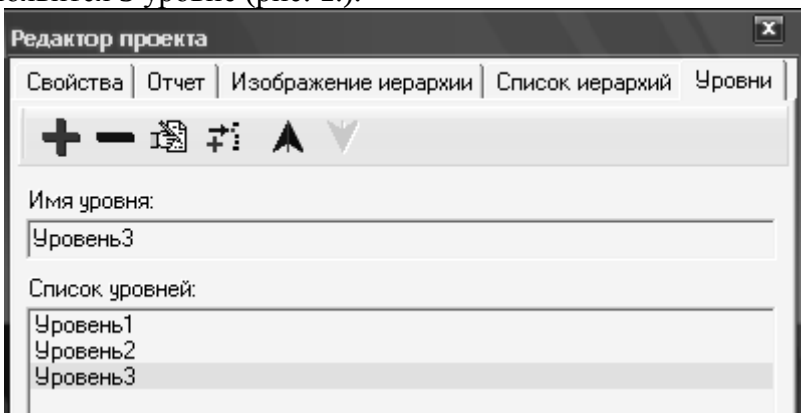


Рис. 1. Окно “Редактор проекта”.

6. Нажимая на каждый из уровней нужно изменить имя каждого из них. Уровень1 – Покупка автомобиля, Уровень2 – Критерии, Уровень3 – Марки авто.

7. Далее нужно нажать на “Покупка автомобиля”, которое находится в списке дважды левой кнопкой мыши.

8. Появится окно “Редактор уровня3 .

9. Нужно щелкнуть на вкладке “Узлы3, и нажимая на красный знак “+” добавить узел, и изменить его имя на “Цель”.

10. Затем нужно нажать ОК и сделать так же с уровнями 2 и 3. Во второй уровень нужно добавить 6 узлов и назвать их критериям условия, в третий уровень нужно добавить минимум 5 узлов и вписать любые марки машин так, чтобы каждой машине соответствовали по крайней мере 2 критерии.

Замечание !!! Для облегчения понимания программы рекомендуем писать только положительные качества предметов цели, то есть если идет речь о выборе машин и о критериях “цена” нужно в критерии добавить узел “высокая цена”. Если авто с низкой ценой между узлом марки авто и узлом “высокая цена” просто не делать связь.

11. Теперь нужно установить связь между каждым узлом. Для этого нужно вызвать окно “Редактор уровня” используя шаги 7 и 8. Окно СППР “Выбор” к установлению связей показаны на рис. 2:

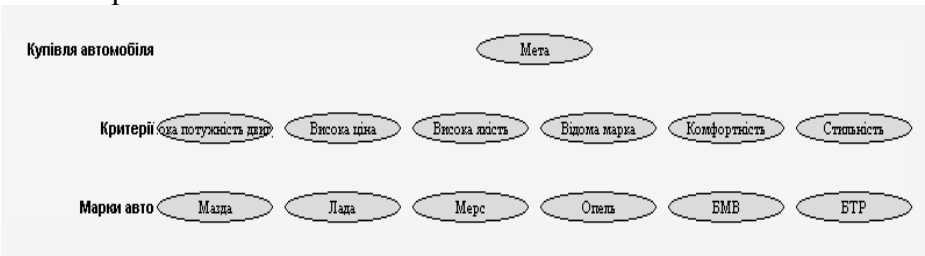


Рис. 2. Окно СППР “Выбор” к установлению связей.

12. Для того чтобы установить связи каждого из узлов нужно дважды щелкнуть на имени узла и в окне “Редактор узла” выбрать вкладку “Связи”.

13. Входных связей первого уровня не будет поэтому нужно выбрать вкладку “Исходящие” и выбрать нажав на стрелку возле выпадающего списка, выбрав в нем критерии.

14. В списке ниже появятся критерии, требуется выбрать все.

15. Так же нужно сделать с уровнем Критерии.

16. Установив необходимые связи получим готовый для расчетов проект (рис. 3.).

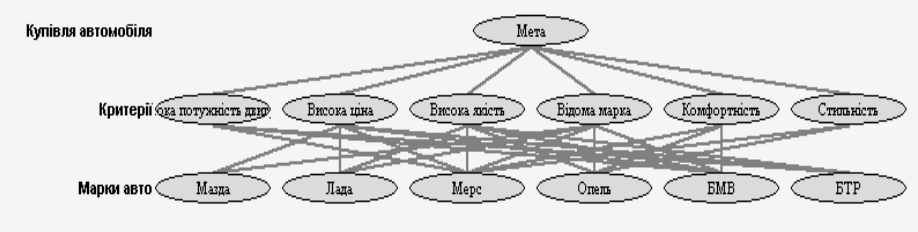


Рис. 3. Окно СППР “Выбор” после установления связей.

Для проверки правильности проделанной работы нужно запустить процесс расчета командой “Проект” – “Расчет”, если все сделано верно, начнется процесс подсчета. В противном случае нужно исправить ошибки. Самая распространенная ошибка-недостаточность связей между узлами, что легко исправить добавив их или заменив другими.

Далее нужно сохранить проект командой “Файл” – “Сохранить”.

Варианты заданий:

1. Создать проект выбора мобильных телефонов
2. Создать проект выбора телевизоров.
3. Создать проект выбора холодильников.
4. Создать проект выбора процессоров.
5. Создать проект выбора ОЗУ.
6. Создать проект выбора Винчестеров.
7. Создать проект выбора колонок.
8. Создать проект выбора магнитол.
9. Создать проект выбора проводов.
10. Создать проект выбора мониторов.
11. Создать проект выбора мешок.
12. Создать проект выбора клавиатур.
13. Создать проект выбора материнских плат.
14. Создать проект выбора принтеров.
15. Создать проект выбора модемов.
16. Создать проект выбора ксероксов.
17. Создать проект выбора стационарных телефонов.
18. Создать проект выбора охлаждающих систем (имеется в виду кулеры или водяное охлаждение).
19. Создать проект выбора тюнеров.
20. Создать проект выбора бесперебойников.

Практическая работа 3. Работа с СППР "Выбор"

Тема: работа с СППР ”Выбор". Расчеты, представление информации, выводы.

Цель занятия:

1. Ознакомиться с основными командами и получить базовые навыки при работе с СППР “Выбор”.

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения:

1. Охарактеризовать принципы работы матрицы парных сравнений.
2. Охарактеризовать окно “Результат вычислений”.
3. Перечислите команды, которые нужно выполнить для изменения вида диаграммы.
4. Для чего нужен индекс согласованности.
5. Для чего нужны веса у исследуемых альтернатив?

Ход работы:

1. Изучить основные аспекты вычислений в СППР “Выбор”.
2. Ознакомиться с интерфейсом и основными командами для проведения вычислений в СППР “Выбор”.
3. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.
4. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

задание

Провести нужные расчеты и оформить выводы с предварительно сделанным проектом выбора машины.

Ход выполнения работы

1. Открыть предыдущий проект командой "Файл” - "Открыть".
2. Командой “Проект” – “Расчет”, вызвать окно, изображенное на рис. 1:

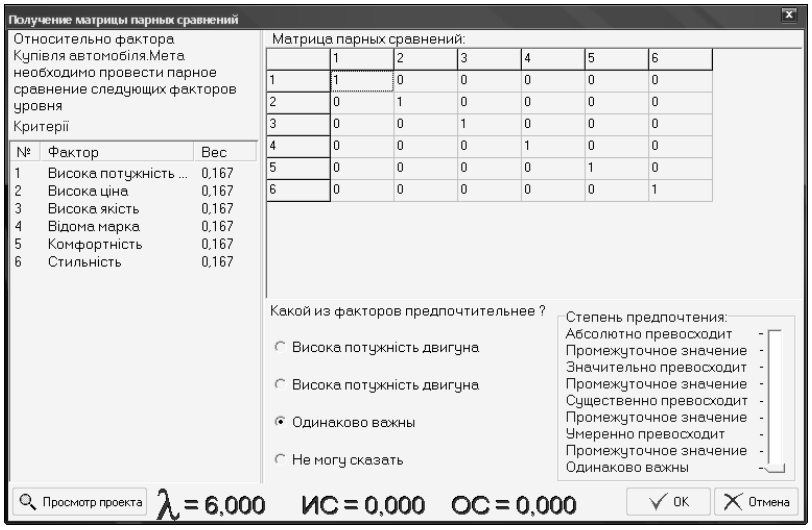


Рис. 1. Окно «Получение матрицы парных сравнений»

3. Данное окно представляет собой нечто вроде тестирующей программы оператором которого является матрица, которую нужно заполнить значениями. Для этого нужно нажимать на каждой ячейке под главной диагональю и отвечать на вопросы. когда

каждое поле будет заполнено нужно нажать ОК программа обработает информацию и выведет следующее окно (рис. 2).

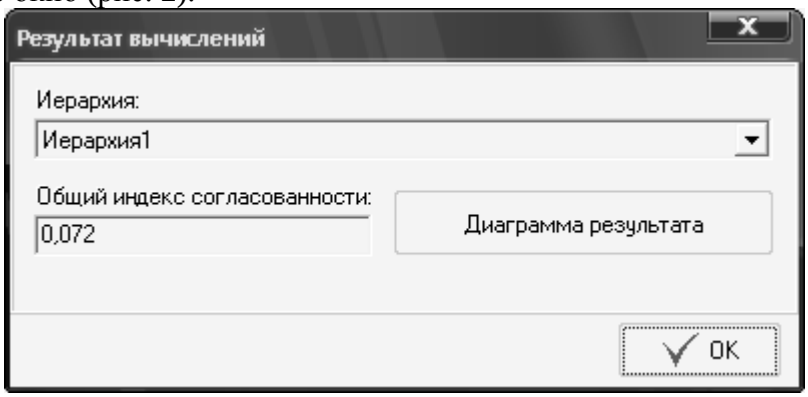


Рис. 2. Окно “Результат вычислений (анализ “Проблема выбора ”)”.

4. Индекс согласованности показывает на то что данные которые Вы ввели не противоречивыми.

5. Далее нужно вывести диаграмму результата нажав на аналогичную клавишу в окне “Результат Вычислений (рис. 3):

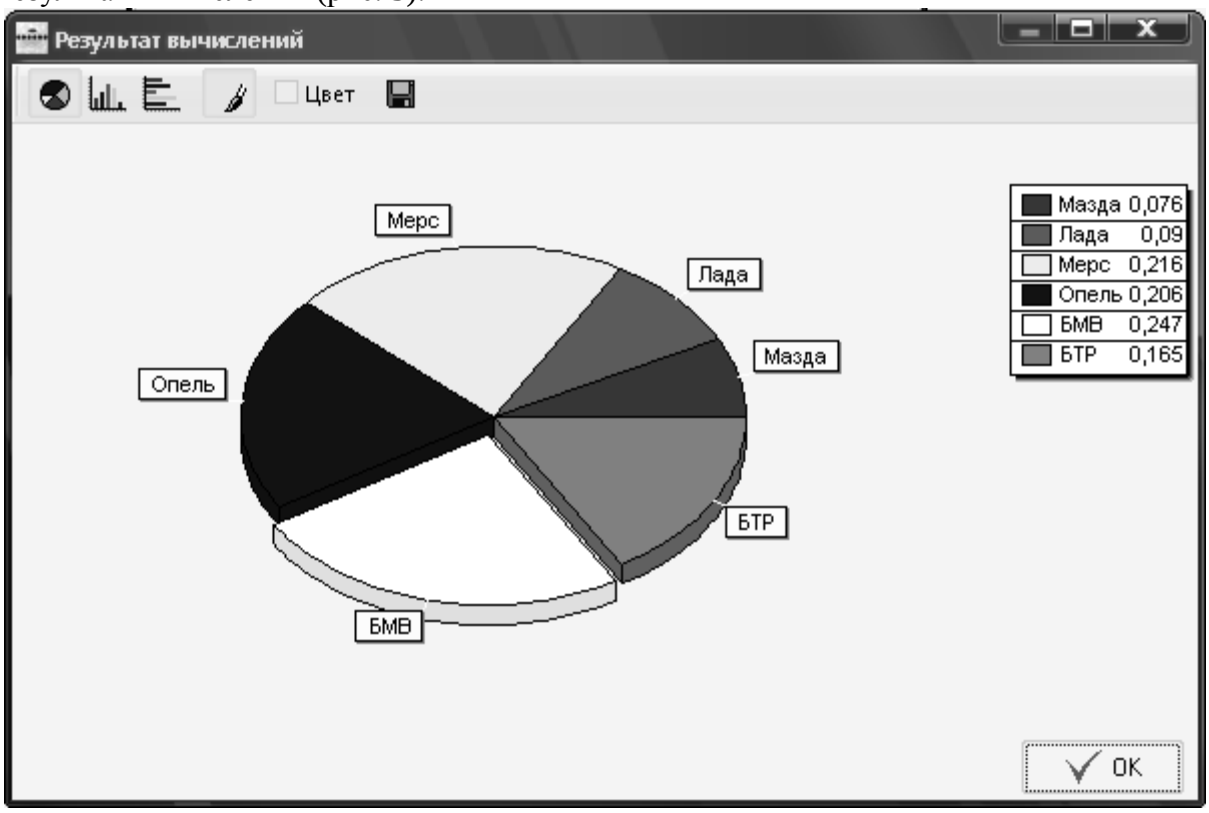


Рис. 3. Окно “Результат Вычислений”

6. В данном окне можно выбрать вид диаграммы, цвет, можно также сохранить ее. Нужно сохранить проект. Диаграмма и расчеты сохраняются вместе с проектом.

Практическая работа 4. Работа с СППР "Выбор"

Тема: работа с СППР”Выбор". Создание проекта типа “Стоимость-эффективность”

Цель занятия:

1.Изучение возможностей проекта типа “Стоимость-эффективность” в программе «Выбор” как элемента моделирования СППР

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения:

1. Чем отличаются проекты “Проблема выбора” и “Стоимость – эффективность”?
2. Для чего нужны уровни критериев и альтернатив?
3. Охарактеризуйте структуру иерархии в СППР”Выбор".
4. Преимущества и недостатки СППР “Выбор”.

Ход работы

1. Проработать теоретический материал, рекомендации по работе с программой.
2. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.
3. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

Постановка задачи и суть проекта типа “Стоимость-эффективность”.

Проекта типа “Стоимость-эффективность”.- это проект, состоящий из двух иерархий, иерархии выгод и иерархии затрат, которые впоследствии необходимо будет между собой ранжировать (сравнивать).

Например, рассматривается задача принятия решения по множеству планов или каких-либо проектов. Классический подход основан на оценке каждого проекта с точки зрения затрат (то есть, сколько необходимо сделать инвестиций для реализации данного проекта) и с точки зрения доходов, которые можно получить при их реализации. Сравнение альтернативных проектов сводится к сравнению объемов доходов из расчета на единицу ресурса (т. е. расходов). Этот метод известен как анализ "Стоимость-эффективность".

Для примера можно предложить задачу по решению целесообразности выбора определенного проекта. Решение задачи ранжирования проектов а, в, С при традиционном подходе может быть сведено в табл. 1.

Таблица 1. Классический подход к задаче "Стоимость-эффективность"

Проект	Расходы	Доходы	Доходы/ расходы	Ранжирования
A	500	1000	2	3
B	250	750	3	1
C	600	1300	2,1	2

При решении этой задачи возникают следующие особенности:

 отношение доходов к затратам, оценивается в стоимостном выражении, по сути не является объективной мерой качества проекта: неясно, как, например, оценивать в

деньгах выгоды и затраты неосязаемых количественно показателей (то есть проблема измерения качественных факторов);

 известно также, что доходы и расходы распределяются по многим сферам - социальным, экономическим, политическим, управленческим и их взаимосвязь влияет на оценку альтернатив.

Применение СППР позволяет снять эти проблемы. В этом случае нужно построить две иерархии: одну для затрат, другую для выгод с одними и теми же альтернативами на нижнем уровне. Таким образом, получают два вектора приоритетов - доходов и расходов. Затем вычисляют отношения доходов к расходам для каждой альтернативы. Наибольшее значение из этих отношений и определяет лучший проект.

В целом решение задач типа "Стоимость-эффективность" происходит в следующем порядке:

а) правильная формулировка цели;

б) построение иерархии выгод (все те же действия, что и в п. 1, но отталкиваться лишь от выгод, не обращает внимания на затраты);

в) построение иерархии издержек (все те же действия, что и в п. 1, но отталкиваться лишь от издержек, не обращающих внимания на выгоды).

Пример решения.

Выбор темы разработки СППР происходит согласно номера студенте в академическом журнале.

Создать СППР для выбора инвестиционного проекта имея такую информацию:

Количество проектов-4.

Критерии оценки (позитивные или выгоды):

7. Надежность;

8. Быстрая окупаемость;

9. Доходность.

Критерии оценки (отрицательные, или расходы):

1. Сложность контроля;

2. Жаль н/с-щу;

3. Удаленность.

Создание проекта

1. Командой “Файл” – “Новый...” – “Простой проект” создать новый проект.

2. Щелкнуть на прямоугольнике по центру окна правой кнопкой мыши, вызвав при этом контекстное меню и в нем выбрать команду “Свойства проекта”.

3. Выбрать вкладку "Уровни".

4. Щелкнуть на знаке “+” 3 раза.

5. В списке появится 3 уровне.

6. Нажимая на каждый из уровней нужно изменить имя каждого из них. Уровень 1-Выбор. проекта, Уровень 2-критерии, Уровень 3-названия проектов (рис. 1).

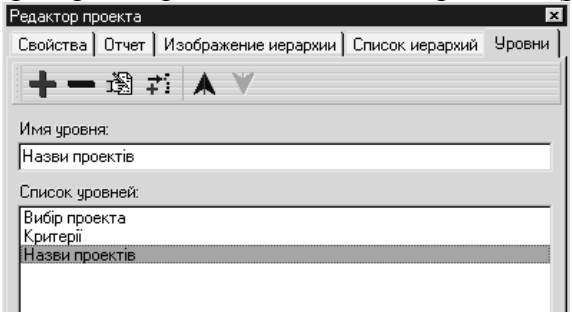


Рис. 1. Диалоговое окно “Редактор проекта”.

7. Далее нужно нажать на “Выбор проекта”, которое находится в списке дважды левой кнопкой мыши.

8. Появится окно “Редактор уровня”.

9. Нужно щелкнуть на вкладке “Узлы”, и нажимая на красный знак “+” добавить узел, и изменить его имя на “Цель”.

10. Затем нужно нажать ОК и сделать так же с уровнями 2 и 3. Во второй уровень нужно добавить 3 узлы и назвать их критериям условия.

11. Теперь нужно установить связь между каждым узлом. Для этого нужно вызвать окно “Редактор уровня” используя шаги 7 и 8.

12. Для того чтобы установить связи каждого из узлов нужно дважды щелкнуть на имени узла и в окне “Редактор узла” выбрать вкладку “Связи”.

13. Входных связей первого уровня не будет поэтому нужно выбрать вкладку “Исходящие” и выбрать нажав на стрелку возле выпадающего списка, выбрав в нем критерии.

14. В списке ниже появятся критерии, требуется выбрать все.

15. Так же нужно сделать с уровнем Критерии.

16. Установив необходимые связи мы выполнили первую часть по созданию проекта типа “Стоимость-эффективность” (рис. 2):

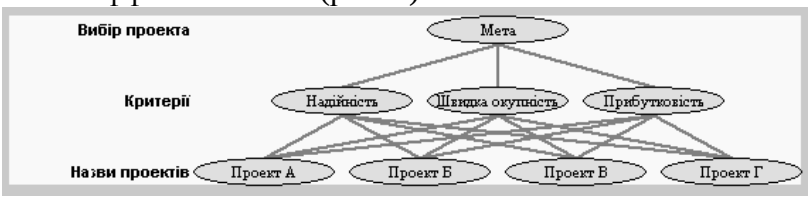


Рис. 2. Схема работы СППР (иерархия предпочтений).

17. Сохраняем проект выполняя команду “Файл” – “Сохранить”. Далее выполняем команду “Проект” – “Иерархия” – “Выбрать иерархию” – “ Иерархия издержек”.

18. Повторяем действия, описанные в пунктах 2-9.

19. Затем нужно нажать ОК и сделать так же с уровнями 2 и 3. Во второй уровень нужно добавить 3 узлы и назвать их критериям из условия (сложность контроля, жаль н/с-щу, удаленность).

20. Повторяем действия, описанные в пунктах 11-15.

21. Сохраняем проект выполняя команду “Файл” – “Сохранить”. В результате получаем схему, изображенную на рис. 3:

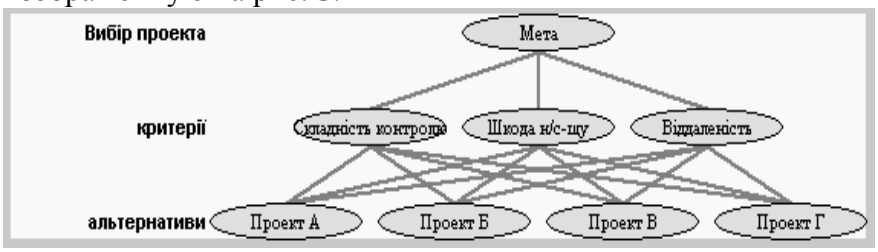


Рис. 3. Схема работы СППР (иерархия потерь)

Для проверки правильности проделанной работы нужно запустить процесс расчета командой “Проект“ – “Расчет”, если все сделано верно, начнется процесс подсчета. В противном случае нужно исправить ошибки. Самая распространенная ошибка-недостаточность связей между узлами, что легко исправить добавив их или заменив другими.

Задания по вариантам:

1. Создать проект выбора мобильных телефонов.
2. Создать проект выбора телевизоров.
3. Создать проект выбора холодильников.
4. Создать проект выбора процессоров.
5. Создать проект выбора ОЗУ.
6. Создать проект выбора Винчестеров.
7. Создать проект выбора колонок.
8. Создать проект выбора магнитол.

9. Создать проект выбора приводов.
10. Создать проект выбора мониторов.
11. Создать проект выбора мышек.
12. Создать проект выбора клавиатур.
13. Создать проект выбора материнских плат.
14. Создать проект выбора принтеров.
15. Создать проект выбора модемов.
16. Создать проект выбора ксероксов.
17. Создать проект выбора стационарных телефонов.
18. Создать проект выбора охлаждающих систем (имеется в виду кулеры или водяное охлаждение).
19. Создать проект выбора тюнеров.
20. Создать проект выбора бесперебойников.

Практическая работа 5. Работа с СППР "Выбор"

Тема: Работа с СППР “Выбор”, средством создания проектов типа “Стоимость-эффективность”. Расчеты, представление информации, выводы

Цель занятия:

1. Изучение возможностей проекта типа “Стоимость-эффективность” в программе “Выбор” как элемента моделирования СППР, завершение проекта предыдущего практического занятия

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения:

1. Отличие матрицы прямых сравнений в проектах “Проблема выбора” и “Стоимость эффективность”.
2. Дайте общую характеристику задачи “Стоимость-эффективность”.
3. Перечислите классы задач, которые может решать проект “Стоимость-эффективность”.
4. Преимущества и недостатки проекта “Стоимость-эффективность”.

Ход работы

1. Изучить основные аспекты вычислений в СППР “Выбор”.
2. Ознакомиться с интерфейсом и основными командами для проведения расчетов в СППР “Выбор”.
3. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.

4. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

Практическое задание.

Произвести нужные вычисления и оформить выводы с предварительно сделанным проектом выбора проекта.

Ход выполнения работы:

1. Предварительный проект командой “Файл” – “Открыть”
2. Командой “Проект” – “Расчет”, вызываем окно, изображенное на рис. 4:

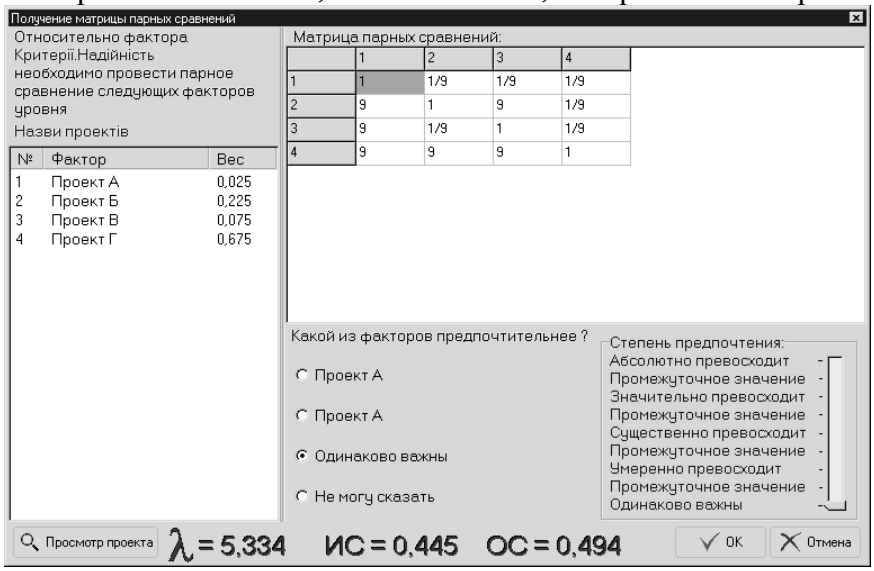


Рис. 4. Окно “Получение матрицы парных сравнений”

3. Данное окно представляет собой нечто вроде тестирующей программы оператором которого является матрица, которую нужно заполнить значениями. Для этого нужно нажимать на каждой ячейке под главной диагональю и отвечать на вопросы. Когда каждое поле будет заполнено нужно нажать ОК программа обработает информацию и выведет следующее окно (рис. 5):

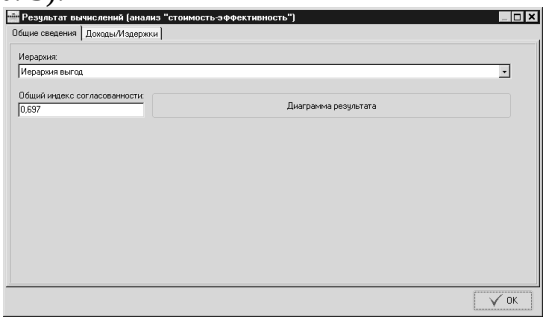


Рис. 5. Окно “Результат вычислений (анализ “стоимость-эффективность”)”.

4. Индекс согласованности показывает на то что данные которые Вы ввели не противоречивыми.

5. Далее нужно вывести диаграмму результата нажав на аналогичную клавишу в окне “Результат Вычислений”. Будет выведена диаграмма иерархии предпочтений (рис. 6). Для вывода иерархии потерь необходимо вернуться в окно “Результат вычислений (анализ “стоимость-эффективность”)” и в поле “Иерархия” выбрать параметр “ Иерархия издержек”.

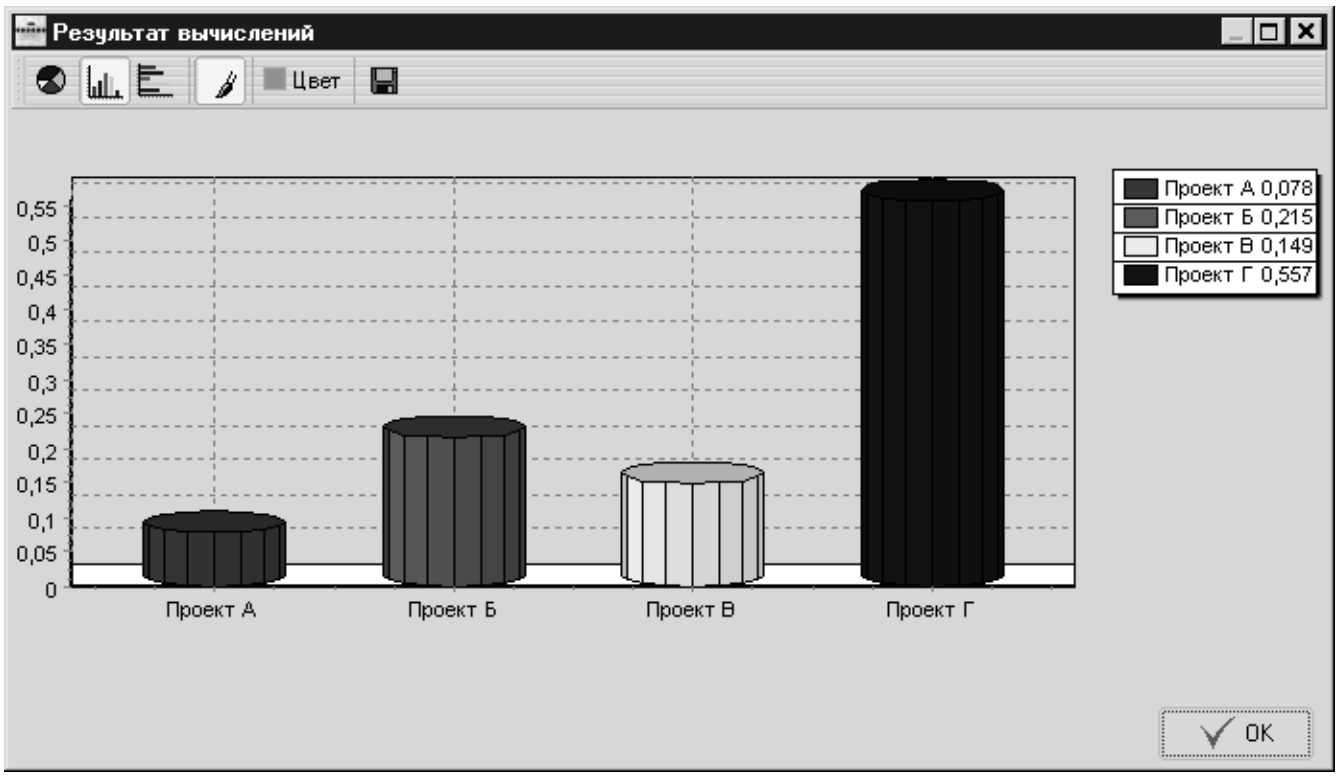


Рис. 6. Окно “Результат вычислений”

6. В данном окне можно выбрать вид диаграммы, цвет, можно также сохранить ее. Нужно сохранить проект. Диаграмма и расчеты сохраняются вместе с проектом. Задание к лабораторной работе брать с предыдущей работы.

Практическая работа 6. Работа с СППР "Выбор"

Тема: Работа с СППР “Выбор”. Расчеты, представление информации, выводы

Цель занятия:

1. Изучение возможностей проекта типа “Поиск наилучшего решения” в программе “Выбор” как элемента моделирования СППР.

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения

1. Теория важности критериев.
2. Свёртка критериев. Однородность критериев.

3. Методы определения качественной важности критериев.
4. Определение количественной важности критериев.
5. Методы определения коэффициентов важности критериев.

Ход работы

1. Изучить основные аспекты вычислений в СППР “Выбор”.

2. Ознакомиться с интерфейсом и основными командами для проведения расчетов в СППР “Выбор”.

3. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.

4. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

С помощью программы «Выбор» попытаемся решить следующую задачу. Нам необходимо произвести отбор кандидатов на освободившуюся должность заместителя начальника отдела информатизации из числа сотрудников отдела. Кандидатов будем оценивать по нескольким критериям:

– стаж работы в организации,
– ответственность,
– образование,
– коммуникабельность.

Мы имеем 4-х претендентов на эту должность.

Таблица 1 – Критерий кандидатов

Ф.И.О.	Критерий			
	Стаж работы	Ответственность	Коммуникабельность	Образование
Иванов	5	Очень ответственный	Коммуникабельный	Высшее техническое
Петров	2	Достаточно ответственный	Замкнут (не коммуникабельный)	Высшее гуманитарное
Сидоров	1	Не ответственный	Очень коммуникабельный	Средне – специальное
Потапов	3	ответственный	Достаточно коммуникабельный	Незаконченное высшее

С помощью программы попытаемся проанализировать, кто из претендентов наиболее подходит.

Сначала нам необходимо ввести в программу данные по критериям и фамилии претендентов. (Рис. 3)

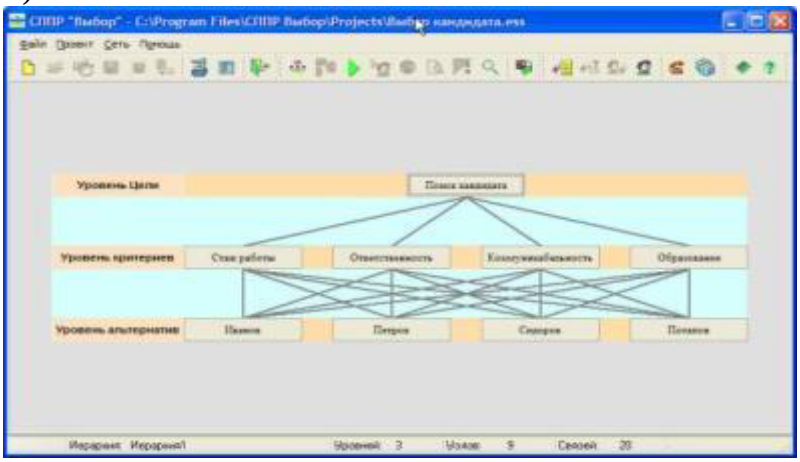


Рисунок 3 – Данные по критериям

Затем мы запускаем выполнение вычислений, где нам необходимо относительно каждого уровня произвести оценку нескольких факторов, тем самым расставив предпочтения (рис. 4)

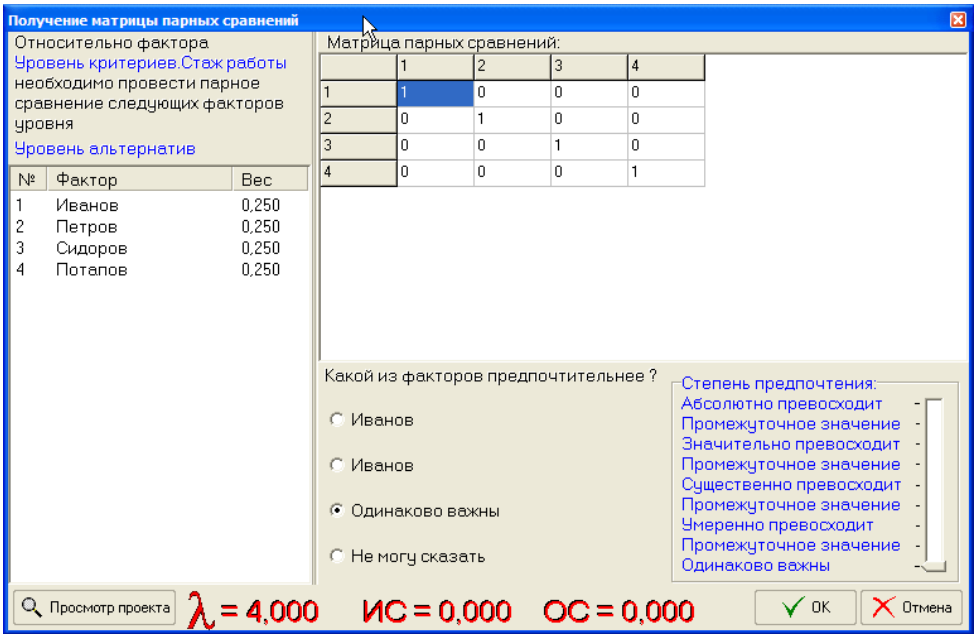


Рисунок 4 – Расчет предпочтений

Для каждого критерия производим оценку, допустим по стажу работы Иванов имеет большее предпочтение так как он проработал на нашем предприятии дольше Петрова. (рис. 5)

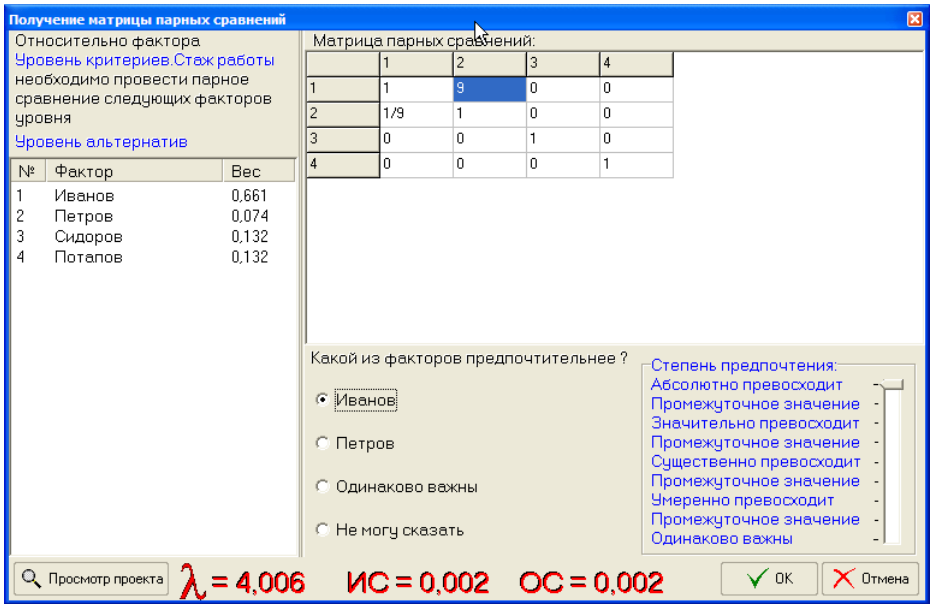


Рисунок 5 – Оценка критериев

Полученные матрицы парных сравнений по критерию стаж работы (рис. 6):

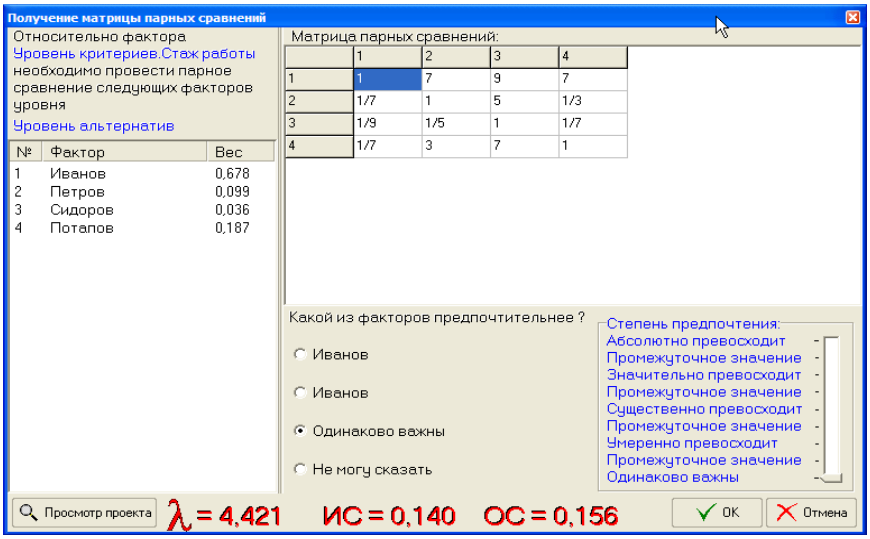


Рисунок 6 – Сравнение по критерию стажа

Иванов проработал дольше всех на предприятии, поэтому ему достается самая высокая оценка, а Сидоров проработал мало, у него самая маленькая.

Полученные матрицы парных сравнений по критерию ответственность (рис. 7):

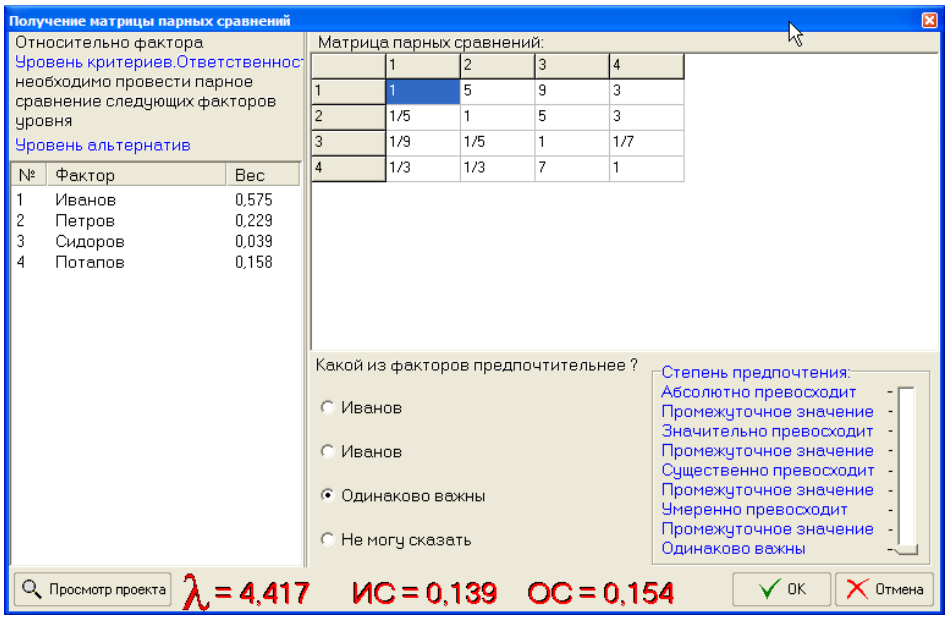


Рисунок 7 – Сравнение по критерию ответственности

Из этой матрицы мы видим, что наибольшая оценка опять достается Иванову, он признан самым ответственным всегда выполняя поручения, Сидоров же наоборот очень часто срывал сроки и относился халатно, поэтому у него самая маленькая оценка.

Полученные матрицы парных сравнений по критерию коммуникабельность (рис. 8):

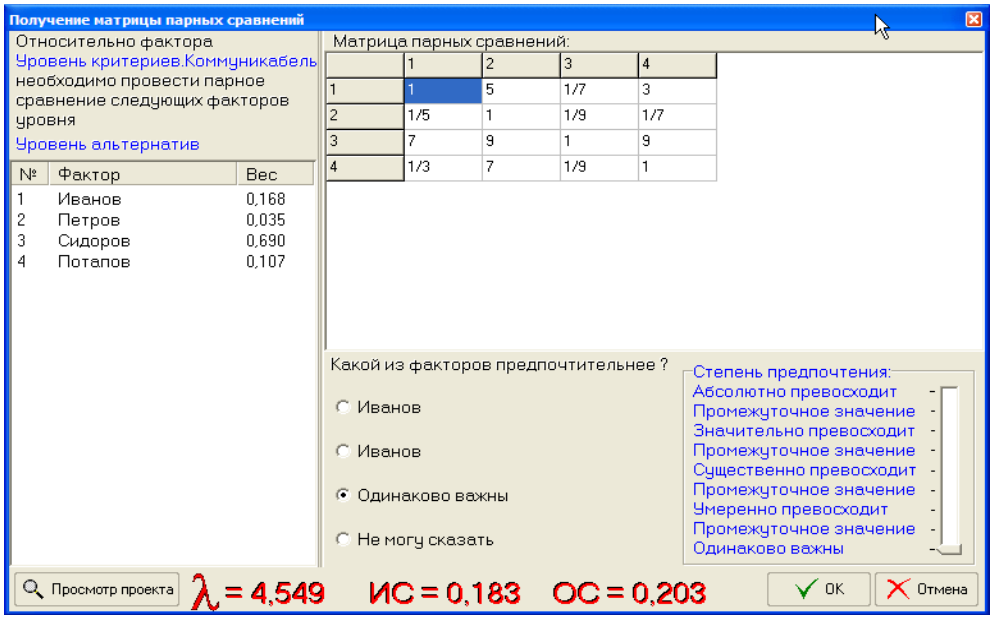


Рисунок 8 – сравнение по критерию коммуникабельности

Здесь мы видим очень интересную картину, по нашим наблюдениям наиболее коммуникабельным признан Сидоров, который по предыдущим критериям был аутсайдером, а вот Петров оказался позади всех, и признан замкнутым человеком.

Комментирую эту матрицу мы должны вспомнить постановку задачи, нам нужен заместитель начальника отдела информатизации, т.е. это должен быть человек хорошо разбирающийся в работе отдела, а значит техник по образованию, но с другой стороны это руководящая должность и возможно человек долго проработавший в отделе хоть и с гуманитарным образованием должен иметь равные шансы. Поэтому как мы видим на рис. 9, Иванову и Петрову проставлены одинаковые оценки, меньше всего у Сидорова т.к. руководитель должен иметь высшее образование.

Полученные матрицы парных сравнений по критерию образование (рис. 9):

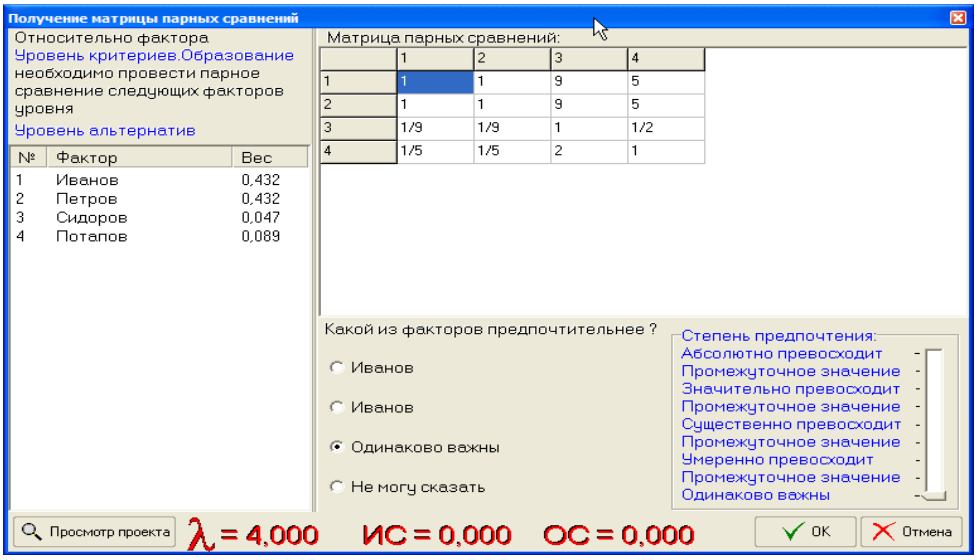


Рисунок 9 – Сравнение по критерию образования

Что для нас является важнее? Стаж, образование, коммуникабельность или ответственность? Думаю, уровняем шансы, нам нужен кандидат в равной степени удовлетворяющий всем параметрам. Расставляем приоритеты поровну. (рис. 10)

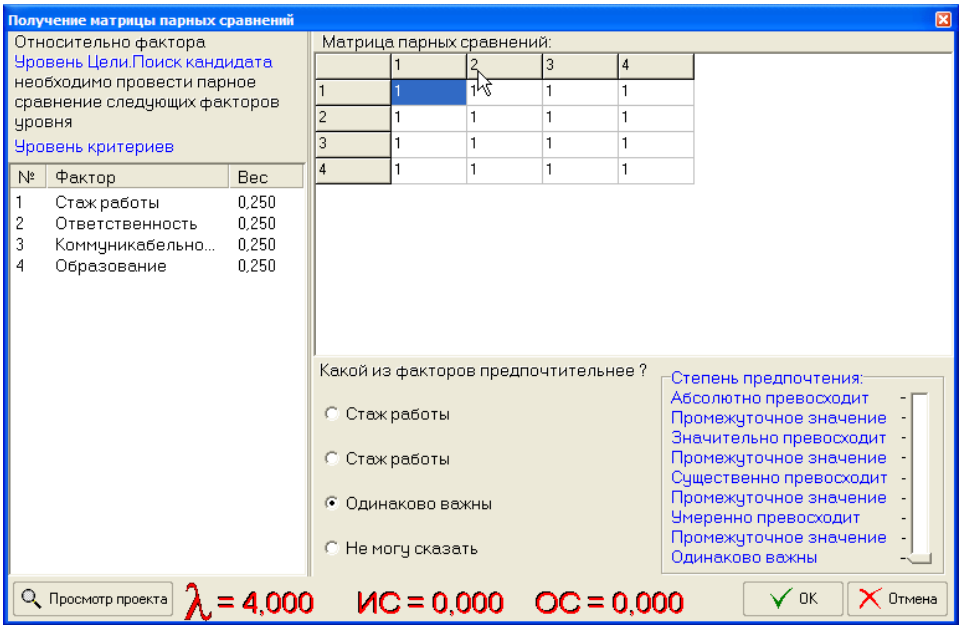


Рисунок 10 – Общие критерий кандидатов

Вычисления закончены, получаем результат (рис. 11):

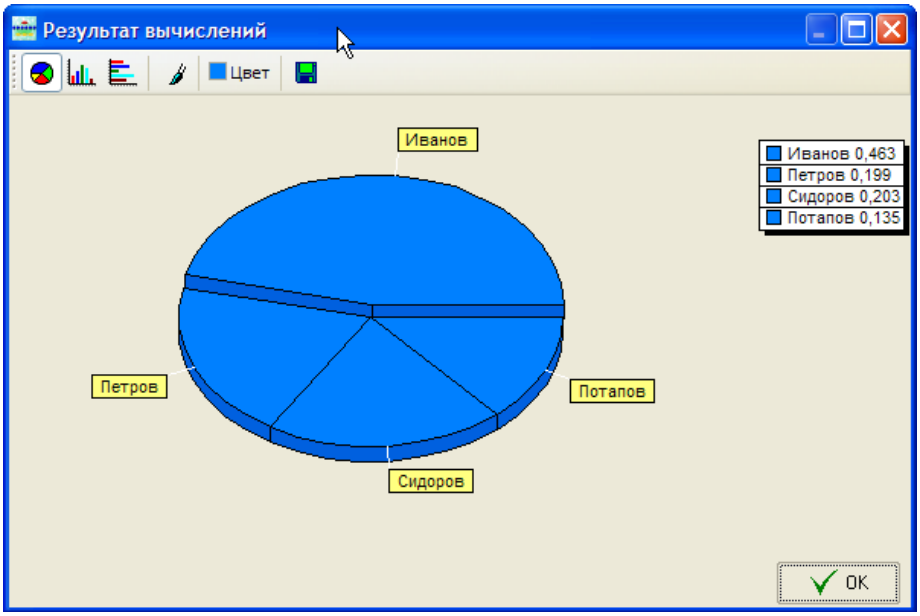


Рисунок 11-Получение результатов

Очевидно, что наиболее подходящим кандидатом на должность заместителя начальника отдела является Иванов.

Задание 1

С помощью программы «Выбор» решить следующую задачу. Нам необходимо произвести отбор кандидатов на освободившуюся должность старосты из числа студентов вашей группы. Кандидатов оценивать по нескольким критериям:

– успеваемость,

– ответственность,

– образование,

– коммуникабельность.

Произвести нужные вычисления и оформить выводы с предварительно сделанным проектом.

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6.1.1. Основная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л1.1	Петров, А. Е.	Математические модели принятия решений: учебно-методическое пособие	Москва: Издательский Дом МИСиС, 2018	http://www.iprbookshop.ru/78572.html
Л1.2	Муромцев, Д. Ю., Шамкин, В. Н.	Методы оптимизации и принятие проектных решений: учебное пособие для магистрантов по направлению 11.04.03	Тамбов: Тамбовский государственный технический университет, ЭБС АСВ, 2015	http://www.iprbookshop.ru/63866.html
Л1.3	Горелик, В. А.	Теория принятия решений: учебное пособие для магистрантов	Москва: Московский педагогический государственный университет, 2016	http://www.iprbookshop.ru/72518.html
6.1.2. Дополнительная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л2.1	Бережная, О. В., Бережная, Е. В.	Методы принятия управленческих решений: учебное пособие	Ставрополь: Северо-Кавказский федеральный университет, 2015	http://www.iprbookshop.ru/62960.html
	Авторы,	Заглавие	Издательство, год	Адрес
Л2.2	Казанская, О. В., Юн, С. Г., Альсова, О. К.	Модели и методы оптимизации. Практикум: учебное пособие	Новосибирск: Новосибирский государственный технический университет, 2012	http://www.iprbookshop.ru/45397.html
6.1.3. Методические разработки				
	Авторы,	Заглавие	Издательство, год	Адрес
Л3.1	Палинчак, Н. Ф., Ярославцева, В. Я.	Системный анализ, оптимизация и принятие решений: методические указания и задания для самостоятельной работы	Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2014	http://www.iprbookshop.ru/55156.html
Л3.2	Артюхин Г. А.	Теория систем и системный анализ. Практикум принятия решений: Учебное пособие	Казань: Казанский государственный архитектурно-строительный университет, ЭБС АСВ, 2016	http://www.iprbookshop.ru/73321.html

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Экономико-математические модели управления»
для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Экономико-математические модели управления содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины - развить системное мышление у обучающихся путем детального анализа подходов к математическому моделированию и сравнительного анализа разных типов моделей. Ознакомить обучающихся с математическими свойствами методов и моделей оптимизации, которые могут использоваться при анализе и решении широкого спектра задач. Выработать у обучающихся навыки проведения численных исследований математических моделей и анализа результатов вычислений. Научить выбирать наиболее перспективное управляющее решение.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Изучив данный курс, студент должен:

Знать:

основы моделирования управленческих решений;

математические модели и информационные технологии процесса поддержки принятия решений;

многокритериальные методы поддержки принятия решений;

основные технологии информационной поддержки процесса поддержки принятия решений;

классификацию систем поддержки принятия решений и особенности используемых инструментальных средств;

современные методы и средства поддержки принятия решений в различных интеллектуальных системах,

принципы их рационального выбора в зависимости от особенностей процесса поддержки принятия решений.

Уметь:

осуществлять постановку конкретных задач поддержки принятия решений, выбирать адекватные математические и инструментальные средства их решения;

решать задачи, связанные с различными этапами подготовки и принятия решений в инструментальных системах

Владеть:

навыками формулирования требований к методам и моделям поддержки принятия решений;

навыками разработки отдельных их элементов;

навыками практического использования моделей и методов поддержки принятия решений;

навыками аналитического обоснования вариантов решений с использованием систем поддержки принятия решений.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическая работа 1. Исследование с помощью системы "Выбор"

Цель занятия:

1. Понятие СППР. Эволюция информационных технологий и информационных систем. Усвоить основные теоретико-информационные понятия учебной дисциплины изучить этапы развития информационной технологии и информационных систем.

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения

1. Дайте определение понятию конкурентные гонки?
2. Что такое информация?
3. Определите содержание СППР?
4. Чем отличаются данные от информации?
5. Основные характеристики данных?

Ход работы

1. Информация и данные.
2. Развитие информационных технологий.
3. Перспективные средства и направления развития информационных систем.
4. Основные понятия систем поддержки принятия решений.

Задача 1. Привести по три определения каждого понятия: “информация”, “данные”, “система поддержки принятия решений” (СППР).

Задача 2. Выделить критерии отбора альтернативных вариантов, которые, по вашему мнению, должны входить в состав СПР выбранной тематики. (например инвестиционные проекты: прибыль, срок окупаемости, и прочее). Также нужно выделить главные и второстепенные критерии. обосновать свой выбор. Тема определяется в соответствии с номером студента в академическом журнале (см темы разработки сппр).

Задача 3. Привести несколько существенных преимуществ применения СППР в выбранной области. Ответ обоснуйте.

Темы разработки сппр:

1. Создание проекта выбора ПК
2. Создание проекта выбора ТВ
3. Создание проекта выбора монитора
4. Создание проекта выбора микроволновой печи
5. Создание проекта выбора автомобиля
6. Создание проекта выбора магнитолы
7. Создание проекта выбора принтера
8. Создание проекта выбора сканера
9. Создание проекта выбора плоттера
10. Создание проекта выбора материнской платы
11. Создание проекта выбора процессора
12. Создание проекта выбора модема
13. Создание проекта выбора мобильного телефона
14. Создание проекта выбора программного обеспечения (по)
15. Создание проекта выбора винчестера (НЖМД)
16. Создание проекта выбора стационарного телефона
17. Создание проекта выбора DVD-проигрывателя
18. Создание проекта выбора интернет-провайдера
19. Создание проекта выбора видеокарты
20. Создание проекта выбора факсимильного аппарата

Практическая работа 2. Работа с СППР "Выбор"

Тема: работа с СППР ”Выбор". Основные функции, приемы и возможности

Цель занятия:

1. Ознакомиться с основными командами и получить базовые навыки при работе с СППР “Выбор”

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения

1. Назовите основные возможности СППР “Выбор”;

2. Перечислите основные команды, которые потребуются для создания нового работоспособного проекта.

3. Перечислите основные возможности окна разработки проектов.

4. Перечислите основные преимущества и недостатки СППР “Выбор”.

Ход работы:

1. Ознакомиться с основным принципами метода анализа иерархий.

2. Изучить основные аспекты интерфейса СППР “Выбор”.

3. Ознакомиться с интерфейсом и основными командами для пользования инструментами доступными в СППР “Выбор”.

4. Познакомиться с основными командами СППР “Выбор”.

5. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.

6. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

Основные аспекты интерфейса СППР "Выбор".

Система поддержки принятия решений (СППР) “Выбор”- аналитическая система, основанная на методе анализа иерархий (МАИ), является простым и удобным средством, которое поможет структурировать проблему, построить набор альтернатив, выделить факторы, характеризующие их, задать значимость этих факторов, оценить альтернативы по каждому из факторов, найти неточности и противоречия в суждениях лица принимающего решение (ЛПР эксперта), проранжировать альтернативы, провести анализ решения и обосновать полученные результаты. Система опирается на математически обоснованный метод анализа иерархий Томаса Саати.

Клиентское применения СППР “Выбор” обладает интуитивным пользовательским интерфейсом. Главное окно-это инструмент работы над проектом, позволяющий просматривать и редактировать выбранную иерархию проекта.

Основные компоненты интерфейса СППР " Выбор”

Панели инструментов.

С помощью панелей инструментов осуществляется быстрый доступ к основным инструментам приложения сосредоточенным в главном меню. Каждому пункту меню соответствует кнопка быстрого запуска на панели инструментов.

Подсказки.

Практически все элементы окон снабжены подсказками, которые появляются при небольшой задержке курсора мыши на необходимом элементе. Более подробная информация обо всех инструментах приложения содержится в помощи, сосредоточенной в главном меню Помощь.

Контекстно-зависимая помощь

СППР “Выбор” обладает мощной справочной системой по каждому инструменту приложения. Более 300 пунктов помощи помогут вам быстрее разобраться как использовать тот или иной элемент приложения. Помощь также предоставляет поиск информации по ключевым словам и фразам.

Помощь по диалоговым окнам

Все диалоговые окна снабжены справочной информацией. Нажмите клавишу F1 для вызова справки по текущему окну. Панели инструментов полностью копируют элементы главного меню. Каждому элементу главного меню соответствует панель инструментов:

* Файл-открытие, сохранение, закрытие проектов, создание новых, настройка приложения, принтера для печати, а также закрытие приложения.

• Проект - работа с иерархиями, произведение расчетов, получение отчетов, редактирование свойств проекта.

* Сеть-отсылка выбранным экспертам текстовых сообщений, проектов, открытие пришедших сообщений, просмотр списка сетевых событий.

* Помощь-вызов справки и окна информации о приложении.

В главном окне приложения для быстрого вызова некоторых инструментов можно использовать следующие горячие клавиши:

Файл

Ctrl + n-новый
F3-Открыть
F2-Сохранить
Ctrl + f2-Сохранить как
F10-Закрыть
F4-настройка принтера
Ctrl+o - Опции

Проект

Ctrl + i - информация о готовности
Ctrl + c-расчет
Ctrl + alt + c - сетевой расчет
Ctrl + r-результаты расчетов
Ctrl + p-просмотр отчета
Ctrl + t-изменить типа
Ctrl+v - Режим просмотра
Shift+Ctrl + P - Свойства

Сеть

F5-текстовое сообщение
F6-послать проект
F10-выбрать экспертов
F9-Эксперты
F11 - Окно сетевых сообщений
F12-окно с пакетами

Помощь

F1-Информация
Ctrl + f1-о программе

В окнах со списками, если фокус ввода находится на списке, то становятся доступными следующие горячие клавиши:

• Insert или "+" - добавление записи.

* Delete или " - " - удаление текущей записи.

Практическое задание.

Создать проект выбора автомобиля имея такую информацию:

Марки автомобилей 5-10 шт.

Критерии оценки:

1. Мощность двигателя (мощный)
2. Цена (Высокая)
3. Качество (Высокое)
4. Ведомость марки (Известная)
5. Комфортность (Комфортная)
6. Стильность (Стильная)

Создание проекта

1. Командой “Файл” – “Новый...” – “Простой проект” создать новый проект.

2. Щелкнуть на прямоугольнике по центру окна правой кнопкой мыши, вызвав при этом контекстное меню и в нем выбрать команду “Свойства проекта”.

3. Выбрать вкладку "Уровни".

4. Щелкнуть на знаке “+” 3 раза.

5. В списке появится 3 уровне (рис. 1.).

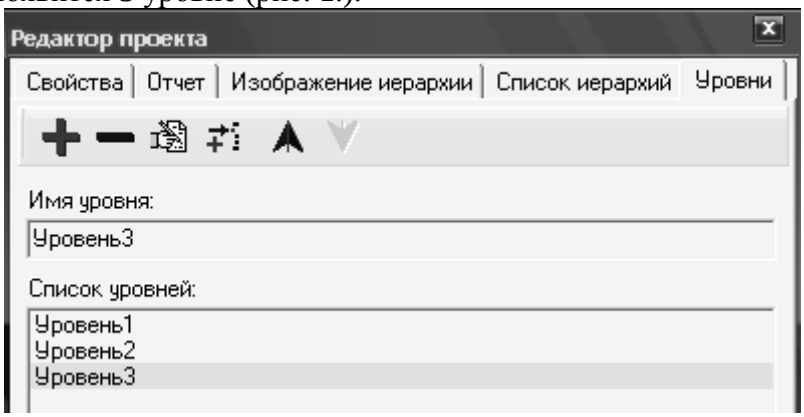


Рис. 1. Окно “Редактор проекта”.

6. Нажимая на каждый из уровней нужно изменить имя каждого из них. Уровень1 – Покупка автомобиля, Уровень2 – Критерии, Уровень3 – Марки авто.

7. Далее нужно нажать на “Покупка автомобиля”, которое находится в списке дважды левой кнопкой мыши.

8. Появится окно “Редактор уровня3 .

9. Нужно щелкнуть на вкладке “Узлы3, и нажимая на красный знак “+” добавить узел, и изменить его имя на “Цель”.

10. Затем нужно нажать ОК и сделать так же с уровнями 2 и 3. Во второй уровень нужно добавить 6 узлов и назвать их критериям условия, в третий уровень нужно добавить минимум 5 узлов и вписать любые марки машин так, чтобы каждой машине соответствовали по крайней мере 2 критерии.

Замечание !!! Для облегчения понимания программы рекомендуем писать только положительные качества предметов цели, то есть если идет речь о выборе машин и о критериях “цена” нужно в критерии добавить узел “высокая цена”. Если авто с низкой ценой между узлом марки авто и узлом “высокая цена” просто не делать связь.

11. Теперь нужно установить связь между каждым узлом. Для этого нужно вызвать окно “Редактор уровня” используя шаги 7 и 8. Окно СППР “Выбор” к установлению связей показаны на рис. 2:

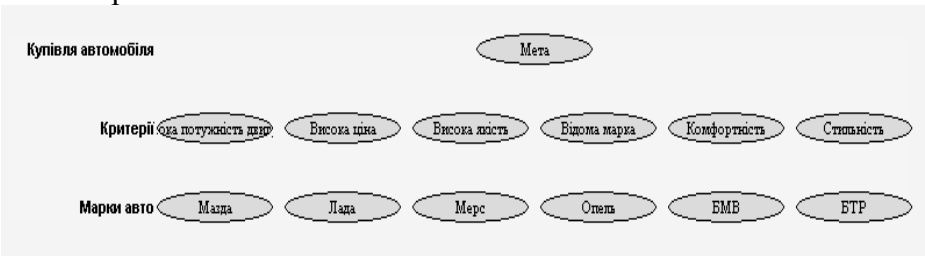


Рис. 2. Окно СППР “Выбор” к установлению связей.

12. Для того чтобы установить связи каждого из узлов нужно дважды щелкнуть на имени узла и в окне “Редактор узла” выбрать вкладку “Связи”.

13. Входных связей первого уровня не будет поэтому нужно выбрать вкладку “Исходящие” и выбрать нажав на стрелку возле выпадающего списка, выбрав в нем критерии.

14. В списке ниже появятся критерии, требуется выбрать все.

15. Так же нужно сделать с уровнем Критерии.

16. Установив необходимые связи получим готовый для расчетов проект (рис. 3.).

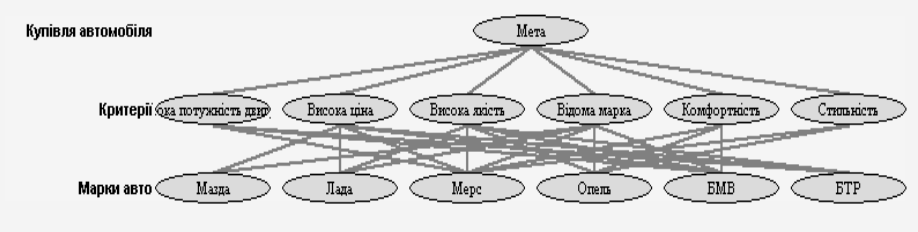


Рис. 3. Окно СППР “Выбор” после установления связей.

Для проверки правильности проделанной работы нужно запустить процесс расчета командой “Проект” – “Расчет”, если все сделано верно, начнется процесс подсчета. В противном случае нужно исправить ошибки. Самая распространенная ошибка-недостаточность связей между узлами, что легко исправить добавив их или заменив другими.

Далее нужно сохранить проект командой “Файл” – “Сохранить”.

Варианты заданий:

1. Создать проект выбора мобильных телефонов
2. Создать проект выбора телевизоров.
3. Создать проект выбора холодильников.
4. Создать проект выбора процессоров.
5. Создать проект выбора ОЗУ.
6. Создать проект выбора Винчестеров.
7. Создать проект выбора колонок.
8. Создать проект выбора магнитол.
9. Создать проект выбора проводов.
10. Создать проект выбора мониторов.
11. Создать проект выбора мешок.
12. Создать проект выбора клавиатур.
13. Создать проект выбора материнских плат.
14. Создать проект выбора принтеров.
15. Создать проект выбора модемов.
16. Создать проект выбора ксероксов.
17. Создать проект выбора стационарных телефонов.
18. Создать проект выбора охлаждающих систем (имеется в виду кулеры или водяное охлаждение).
19. Создать проект выбора тюнеров.
20. Создать проект выбора бесперебойников.

Практическая работа 3. Работа с СППР "Выбор"

Тема: работа с СППР ”Выбор". Расчеты, представление информации, выводы.

Цель занятия:

1. Ознакомиться с основными командами и получить базовые навыки при работе с СППР “Выбор”.

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения:

1. Охарактеризовать принципы работы матрицы парных сравнений.
2. Охарактеризовать окно “Результат вычислений”.
3. Перечислите команды, которые нужно выполнить для изменения вида диаграммы.
4. Для чего нужен индекс согласованности.
5. Для чего нужны веса у исследуемых альтернатив?

Ход работы:

1. Изучить основные аспекты вычислений в СППР “Выбор”.
2. Ознакомиться с интерфейсом и основными командами для проведения вычислений в СППР “Выбор”.
3. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.
4. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

задание

Провести нужные расчеты и оформить выводы с предварительно сделанным проектом выбора машины.

Ход выполнения работы

1. Открыть предыдущий проект командой "Файл” - "Открыть".
2. Командой “Проект” – “Расчет”, вызвать окно, изображенное на рис. 1:

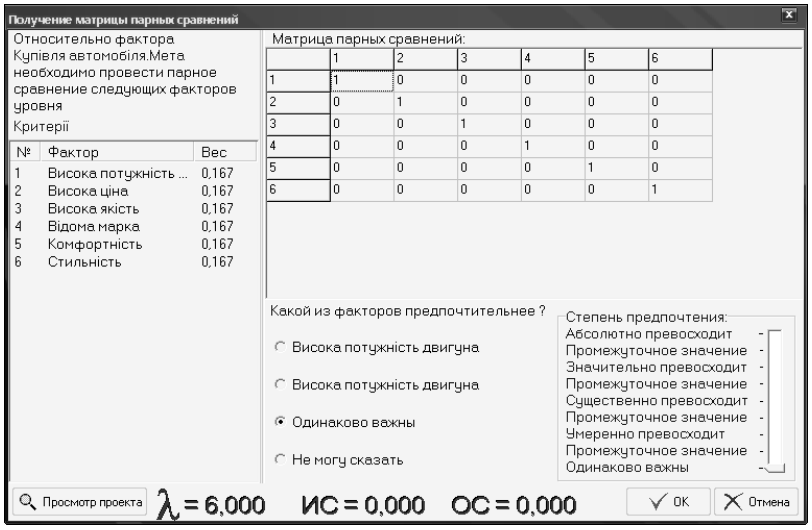


Рис. 1. Окно «Получение матрицы парных сравнений»

3. Данное окно представляет собой нечто вроде тестирующей программы оператором которого является матрица, которую нужно заполнить значениями. Для этого нужно нажимать на каждой ячейке под главной диагональю и отвечать на вопросы. когда

каждое поле будет заполнено нужно нажать ОК программа обработает информацию и выведет следующее окно (рис. 2).

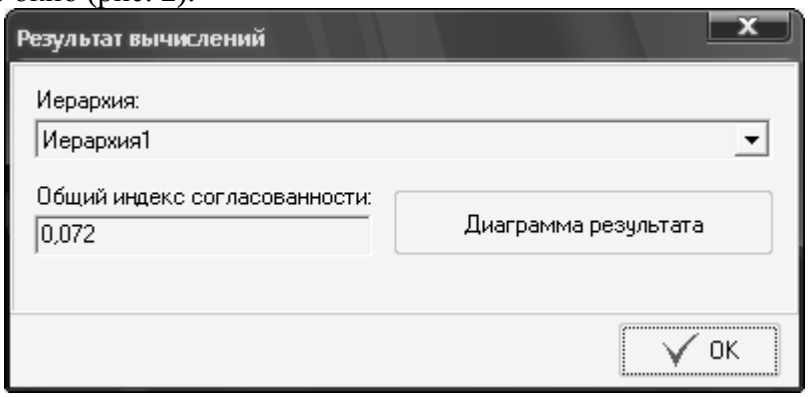


Рис. 2. Окно “Результат вычислений (анализ “Проблема выбора ”)”.

4. Индекс согласованности показывает на то что данные которые Вы ввели не противоречивыми.

5. Далее нужно вывести диаграмму результата нажав на аналогичную клавишу в окне “Результат Вычислений (рис. 3):

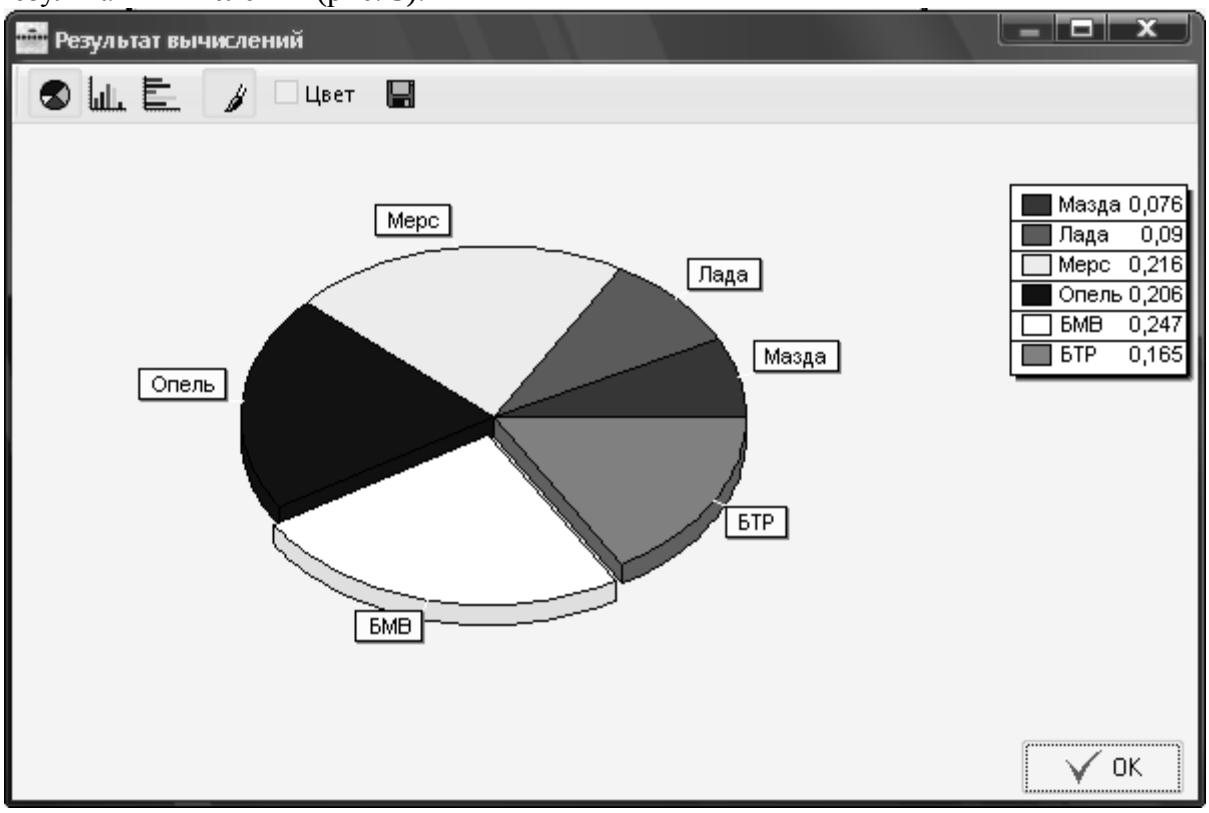


Рис. 3. Окно “Результат Вычислений”

6. В данном окне можно выбрать вид диаграммы, цвет, можно также сохранить ее. Нужно сохранить проект. Диаграмма и расчеты сохраняются вместе с проектом.

Практическая работа 4. Работа с СППР "Выбор"

Тема: работа с СППР”Выбор". Создание проекта типа “Стоимость-эффективность”

Цель занятия:

1.Изучение возможностей проекта типа “Стоимость-эффективность” в программе «Выбор” как элемента моделирования СППР

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения:

1. Чем отличаются проекты “Проблема выбора” и “Стоимость – эффективность”?
2. Для чего нужны уровни критериев и альтернатив?
3. Охарактеризуйте структуру иерархии в СППР”Выбор".
4. Преимущества и недостатки СППР “Выбор”.

Ход работы

1. Проработать теоретический материал, рекомендации по работе с программой.
2. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.
3. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

Постановка задачи и суть проекта типа “Стоимость-эффективность”.

Проекта типа “Стоимость-эффективность”.- это проект, состоящий из двух иерархий, иерархии выгод и иерархии затрат, которые впоследствии необходимо будет между собой ранжировать (сравнивать).

Например, рассматривается задача принятия решения по множеству планов или каких-либо проектов. Классический подход основан на оценке каждого проекта с точки зрения затрат (то есть, сколько необходимо сделать инвестиций для реализации данного проекта) и с точки зрения доходов, которые можно получить при их реализации. Сравнение альтернативных проектов сводится к сравнению объемов доходов из расчета на единицу ресурса (т. е. расходов). Этот метод известен как анализ "Стоимость-эффективность".

Для примера можно предложить задачу по решению целесообразности выбора определенного проекта. Решение задачи ранжирования проектов а, в, С при традиционном подходе может быть сведено в табл. 1.

Таблица 1. Классический подход к задаче "Стоимость-эффективность"

Проект	Расходы	Доходы	Доходы/ расходы	Ранжирования
A	500	1000	2	3
B	250	750	3	1
C	600	1300	2,1	2

При решении этой задачи возникают следующие особенности:

 отношение доходов к затратам, оценивается в стоимостном выражении, по сути не является объективной мерой качества проекта: неясно, как, например, оценивать в

деньгах выгоды и затраты неосязаемых количественно показателей (то есть проблема измерения качественных факторов);

 известно также, что доходы и расходы распределяются по многим сферам - социальным, экономическим, политическим, управленческим и их взаимосвязь влияет на оценку альтернатив.

Применение СППР позволяет снять эти проблемы. В этом случае нужно построить две иерархии: одну для затрат, другую для выгод с одними и теми же альтернативами на нижнем уровне. Таким образом, получают два вектора приоритетов - доходов и расходов. Затем вычисляют отношения доходов к расходам для каждой альтернативы. Наибольшее значение из этих отношений и определяет лучший проект.

В целом решение задач типа "Стоимость-эффективность" происходит в следующем порядке:

а) правильная формулировка цели;

б) построение иерархии выгод (все те же действия, что и в п. 1, но отталкиваться лишь от выгод, не обращает внимания на затраты);

в) построение иерархии издержек (все те же действия, что и в п. 1, но отталкиваться лишь от издержек, не обращающих внимания на выгоды).

Пример решения.

Выбор темы разработки СППР происходит согласно номера студенте в академическом журнале.

Создать СППР для выбора инвестиционного проекта имея такую информацию:

Количество проектов-4.

Критерии оценки (позитивные или выгоды):

7. Надежность;

8. Быстрая окупаемость;

9. Доходность.

Критерии оценки (отрицательные, или расходы):

1. Сложность контроля;

2. Жаль н/с-щу;

3. Удаленность.

Создание проекта

1. Командой “Файл” – “Новый...” – “Простой проект” создать новый проект.

2. Щелкнуть на прямоугольнике по центру окна правой кнопкой мыши, вызвав при этом контекстное меню и в нем выбрать команду “Свойства проекта”.

3. Выбрать вкладку "Уровни".

4. Щелкнуть на знаке “+” 3 раза.

5. В списке появится 3 уровне.

6. Нажимая на каждый из уровней нужно изменить имя каждого из них. Уровень 1-Выбор. проекта, Уровень 2-критерии, Уровень 3-названия проектов (рис. 1).

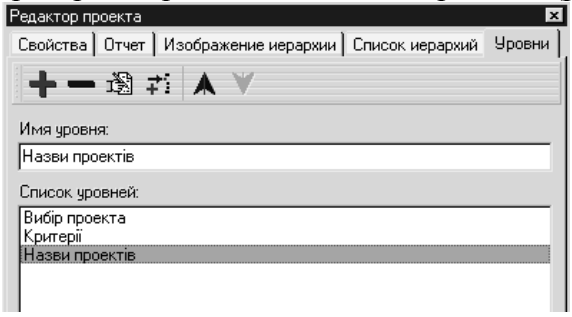


Рис. 1. Диалоговое окно “Редактор проекта”.

7. Далее нужно нажать на “Выбор проекта”, которое находится в списке дважды левой кнопкой мыши.

8. Появится окно “Редактор уровня”.

9. Нужно щелкнуть на вкладке “Узлы”, и нажимая на красный знак “+” добавить узел, и изменить его имя на “Цель”.

10. Затем нужно нажать ОК и сделать так же с уровнями 2 и 3. Во второй уровень нужно добавить 3 узлы и назвать их критериям условия.

11. Теперь нужно установить связь между каждым узлом. Для этого нужно вызвать окно “Редактор уровня” используя шаги 7 и 8.

12. Для того чтобы установить связи каждого из узлов нужно дважды щелкнуть на имени узла и в окне “Редактор узла” выбрать вкладку “Связи”.

13. Входных связей первого уровня не будет поэтому нужно выбрать вкладку “Исходящие” и выбрать нажав на стрелку возле выпадающего списка, выбрав в нем критерии.

14. В списке ниже появятся критерии, требуется выбрать все.

15. Так же нужно сделать с уровнем Критерии.

16. Установив необходимые связи мы выполнили первую часть по созданию проекта типа “Стоимость-эффективность” (рис. 2):

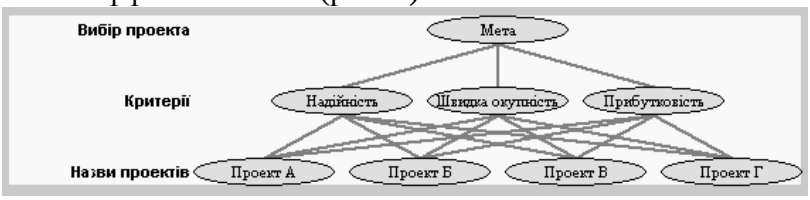


Рис. 2. Схема работы СППР (иерархия предпочтений).

17. Сохраняем проект выполняя команду “Файл” – “Сохранить”. Далее выполняем команду “Проект” – “Иерархия” – “Выбрать иерархию” – “ Иерархия издержек”.

18. Повторяем действия, описанные в пунктах 2-9.

19. Затем нужно нажать ОК и сделать так же с уровнями 2 и 3. Во второй уровень нужно добавить 3 узлы и назвать их критериям из условия (сложность контроля, жаль н/с-щу, удаленность).

20. Повторяем действия, описанные в пунктах 11-15.

21. Сохраняем проект выполняя команду “Файл” – “Сохранить”. В результате получаем схему, изображенную на рис. 3:

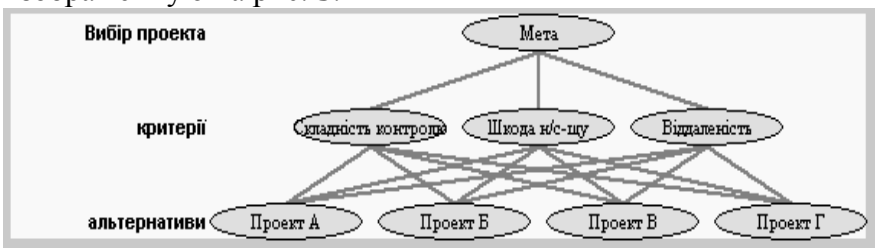


Рис. 3. Схема работы СППР (иерархия потерь)

Для проверки правильности проделанной работы нужно запустить процесс расчета командой “Проект“ – “Расчет”, если все сделано верно, начнется процесс подсчета. В противном случае нужно исправить ошибки. Самая распространенная ошибка-недостаточность связей между узлами, что легко исправить добавив их или заменив другими.

Задания по вариантам:

1. Создать проект выбора мобильных телефонов.
2. Создать проект выбора телевизоров.
3. Создать проект выбора холодильников.
4. Создать проект выбора процессоров.
5. Создать проект выбора ОЗУ.
6. Создать проект выбора Винчестеров.
7. Создать проект выбора колонок.
8. Создать проект выбора магнитол.

9. Создать проект выбора приводов.
10. Создать проект выбора мониторов.
11. Создать проект выбора мышек.
12. Создать проект выбора клавиатур.
13. Создать проект выбора материнских плат.
14. Создать проект выбора принтеров.
15. Создать проект выбора модемов.
16. Создать проект выбора ксероксов.
17. Создать проект выбора стационарных телефонов.
18. Создать проект выбора охлаждающих систем (имеется в виду кулеры или водяное охлаждение).
19. Создать проект выбора тюнеров.
20. Создать проект выбора бесперебойников.

Практическая работа 5. Работа с СППР "Выбор"

Тема: Работа с СППР “Выбор”, средством создания проектов типа “Стоимость-эффективность”. Расчеты, представление информации, выводы

Цель занятия:

1. Изучение возможностей проекта типа “Стоимость-эффективность” в программе “Выбор” как элемента моделирования СППР, завершение проекта предыдущего практического занятия

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения:

1. Отличие матрицы прямых сравнений в проектах “Проблема выбора” и “Стоимость эффективность”.
2. Дайте общую характеристику задачи “Стоимость-эффективность”.
3. Перечислите классы задач, которые может решать проект “Стоимость-эффективность”.
4. Преимущества и недостатки проекта “Стоимость-эффективность”.

Ход работы

1. Изучить основные аспекты вычислений в СППР “Выбор”.
2. Ознакомиться с интерфейсом и основными командами для проведения расчетов в СППР “Выбор”.
3. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.

4. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

Практическое задание.

Произвести нужные вычисления и оформить выводы с предварительно сделанным проектом выбора проекта.

Ход выполнения работы:

1. Предварительный проект командой “Файл” – “Открыть”
2. Командой “Проект” – “Расчет”, вызываем окно, изображенное на рис. 4:

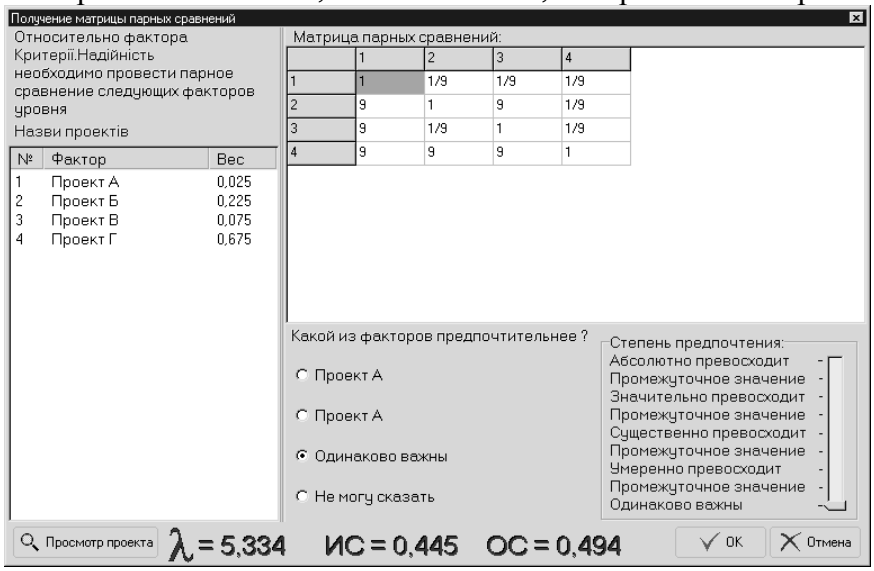


Рис. 4. Окно “Получение матрицы парных сравнений”

3. Данное окно представляет собой нечто вроде тестирующей программы оператором которого является матрица, которую нужно заполнить значениями. Для этого нужно нажимать на каждой ячейке под главной диагональю и отвечать на вопросы. Когда каждое поле будет заполнено нужно нажать ОК программа обработает информацию и выведет следующее окно (рис. 5):

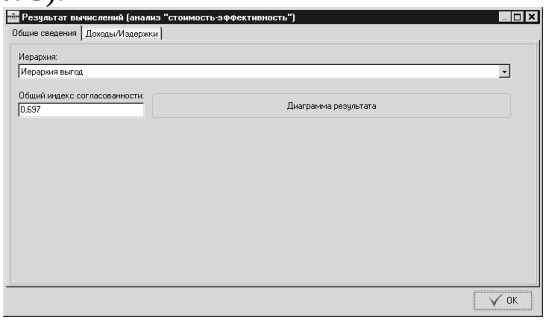


Рис. 5. Окно “Результат вычислений (анализ “стоимость-эффективность”)”.

4. Индекс согласованности показывает на то что данные которые Вы ввели не противоречивыми.

5. Далее нужно вывести диаграмму результата нажав на аналогичную клавишу в окне “Результат Вычислений”. Будет выведена диаграмма иерархии предпочтений (рис. 6). Для вывода иерархии потерь необходимо вернуться в окно “Результат вычислений (анализ “стоимость-эффективность”)” и в поле “Иерархия” выбрать параметр “ Иерархия издержек”.

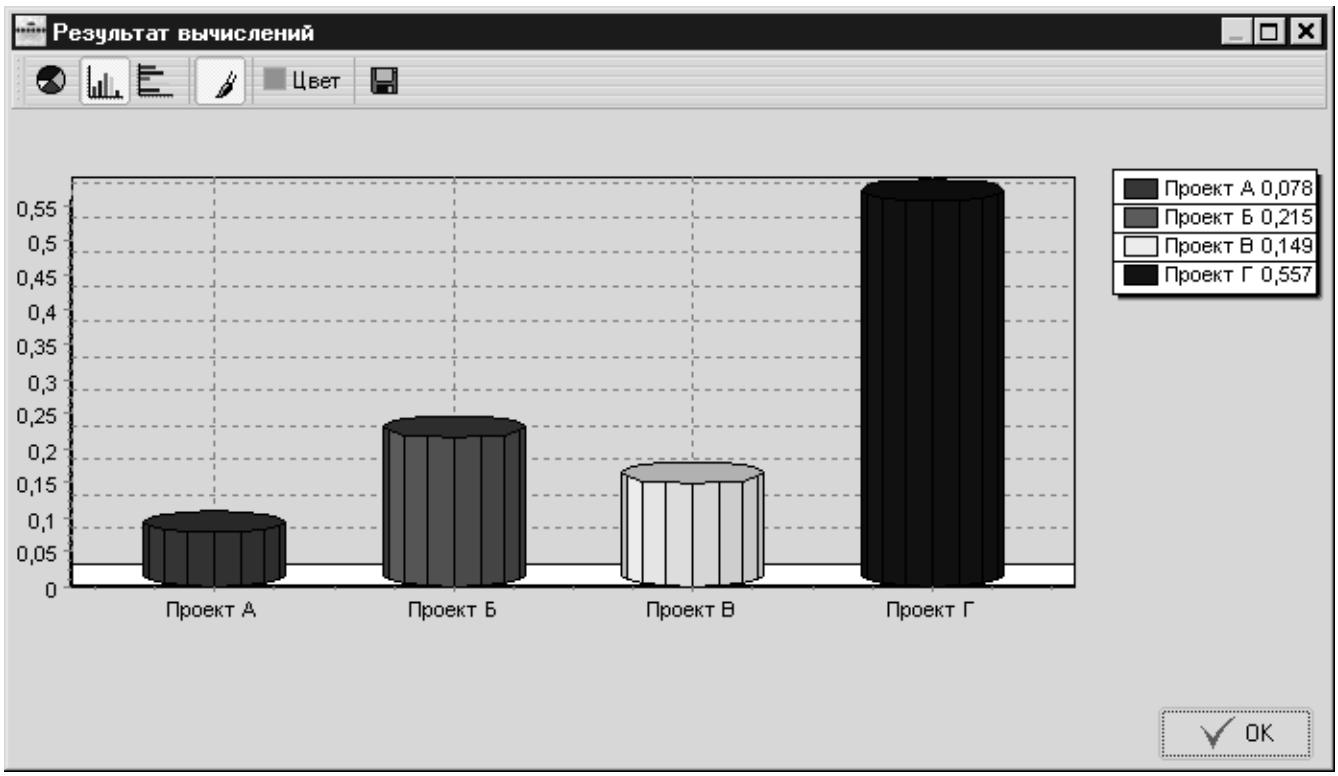


Рис. 6. Окно “Результат вычислений”

6. В данном окне можно выбрать вид диаграммы, цвет, можно также сохранить ее. Нужно сохранить проект. Диаграмма и расчеты сохраняются вместе с проектом. Задание к лабораторной работе брать с предыдущей работы.

Практическая работа 6. Работа с СППР "Выбор"

Тема: Работа с СППР “Выбор”. Расчеты, представление информации, выводы

Цель занятия:

1. Изучение возможностей проекта типа “Поиск наилучшего решения” в программе “Выбор” как элемента моделирования СППР.

2. Формирование компетенций:

УК-2: Способен управлять проектом на всех этапах его жизненного цикла;

УК-2.3: Объясняет цели и формулирует задачи, связанные с подготовкой и реализацией проекта, управляет проектом на всех этапах его жизненного цикла.

ОПК_4: Способен применять на практике новые научные принципы и методы исследований

ОПК-4.2: Применяет на практике новые методы исследований

ОПК-7: Способен разрабатывать и применять математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

ОПК-7/2: Разрабатывает и применяет математические модели процессов и объектов при решении задач анализа и синтеза распределенных информационных систем и систем поддержки принятия решений

Вопросы для обсуждения

1. Теория важности критериев.
2. Свёртка критериев. Однородность критериев.

3. Методы определения качественной важности критериев.
4. Определение количественной важности критериев.
5. Методы определения коэффициентов важности критериев.

Ход работы

1. Изучить основные аспекты вычислений в СППР “Выбор”.

2. Ознакомиться с интерфейсом и основными командами для проведения расчетов в СППР “Выбор”.

3. Выполнить дополнительное задание лабораторной работы и дать ответы на дополнительные вопросы.

4. Оформить отчет по лабораторной работе, который включает выводы относительно возможностей использования средств анализа данных в СППР “Выбор” в СППР, описание примеров, иллюстративный материал.

С помощью программы «Выбор» попытаемся решить следующую задачу. Нам необходимо произвести отбор кандидатов на освободившуюся должность заместителя начальника отдела информатизации из числа сотрудников отдела. Кандидатов будем оценивать по нескольким критериям:

– стаж работы в организации,
– ответственность,
– образование,
– коммуникабельность.

Мы имеем 4-х претендентов на эту должность.

Таблица 1 – Критерий кандидатов

Ф.И.О.	Критерий			
	Стаж работы	Ответственность	Коммуникабельность	Образование
Иванов	5	Очень ответственный	Коммуникабельный	Высшее техническое
Петров	2	Достаточно ответственный	Замкнут (не коммуникабельный)	Высшее гуманитарное
Сидоров	1	Не ответственный	Очень коммуникабельный	Средне – специальное
Потапов	3	ответственный	Достаточно коммуникабельный	Незаконченное высшее

С помощью программы попытаемся проанализировать, кто из претендентов наиболее подходит.

Сначала нам необходимо ввести в программу данные по критериям и фамилии претендентов. (Рис. 3)

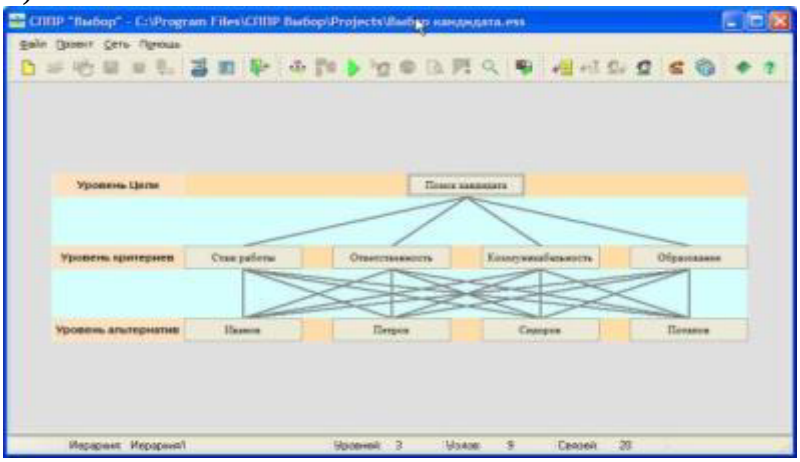


Рисунок 3 – Данные по критериям

Затем мы запускаем выполнение вычислений, где нам необходимо относительно каждого уровня произвести оценку нескольких факторов, тем самым расставив предпочтения (рис. 4)

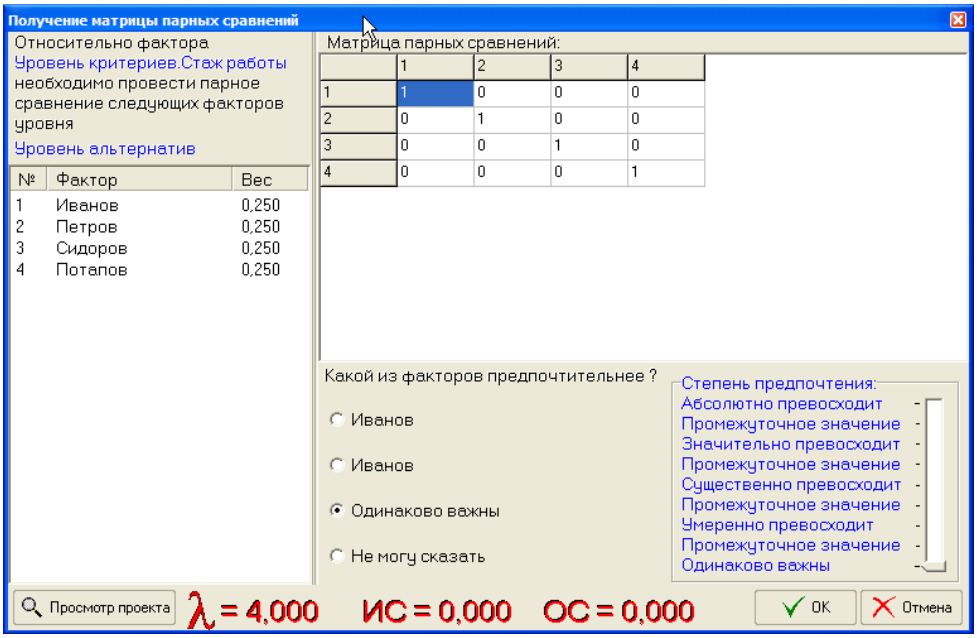


Рисунок 4 – Расчет предпочтений

Для каждого критерия производим оценку, допустим по стажу работы Иванов имеет большее предпочтение так как он проработал на нашем предприятии дольше Петрова. (рис. 5)

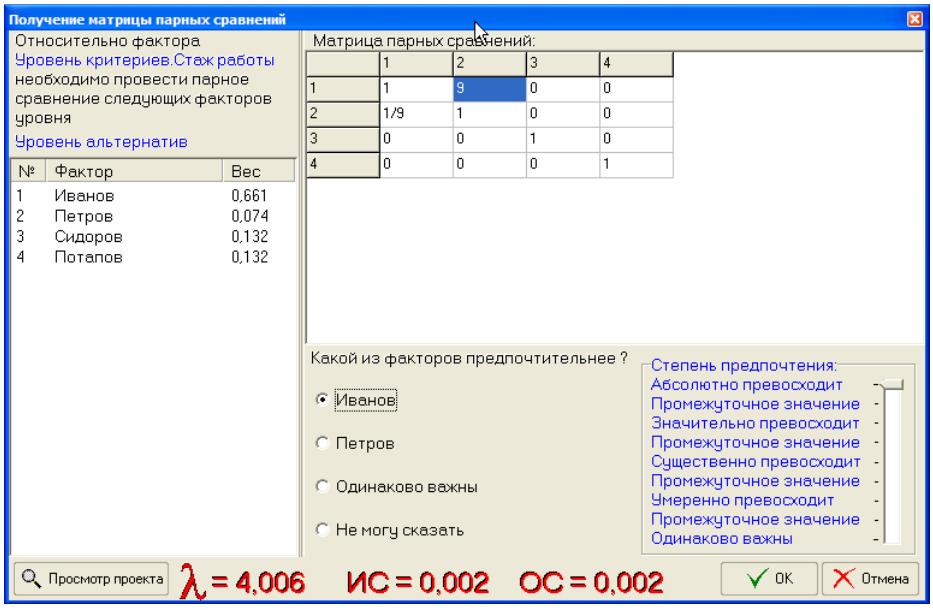


Рисунок 5 – Оценка критериев

Полученные матрицы парных сравнений по критерию стаж работы (рис. 6):

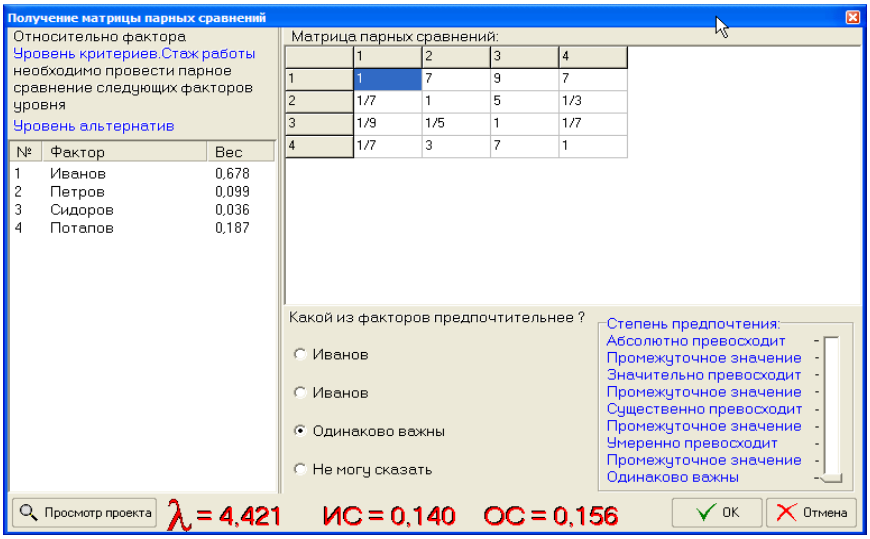


Рисунок 6 – Сравнение по критерию стажа

Иванов проработал дольше всех на предприятии, поэтому ему достается самая высокая оценка, а Сидоров проработал мало, у него самая маленькая.

Полученные матрицы парных сравнений по критерию ответственность (рис. 7):

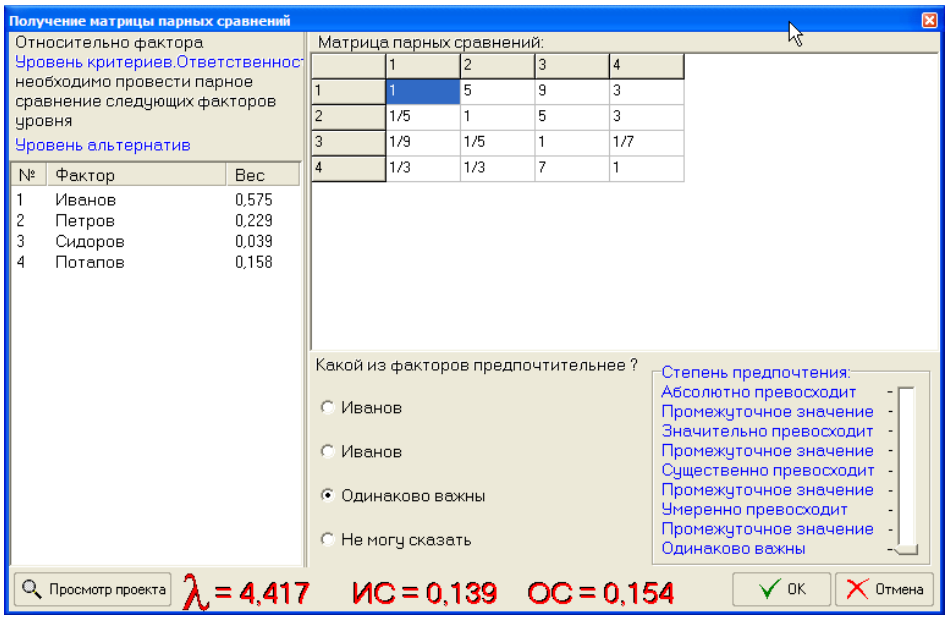


Рисунок 7 – Сравнение по критерию ответственности

Из этой матрицы мы видим, что наибольшая оценка опять достается Иванову, он признан самым ответственным всегда выполняя поручения, Сидоров же наоборот очень часто срывал сроки и относился халатно, поэтому у него самая маленькая оценка.

Полученные матрицы парных сравнений по критерию коммуникабельность (рис. 8):

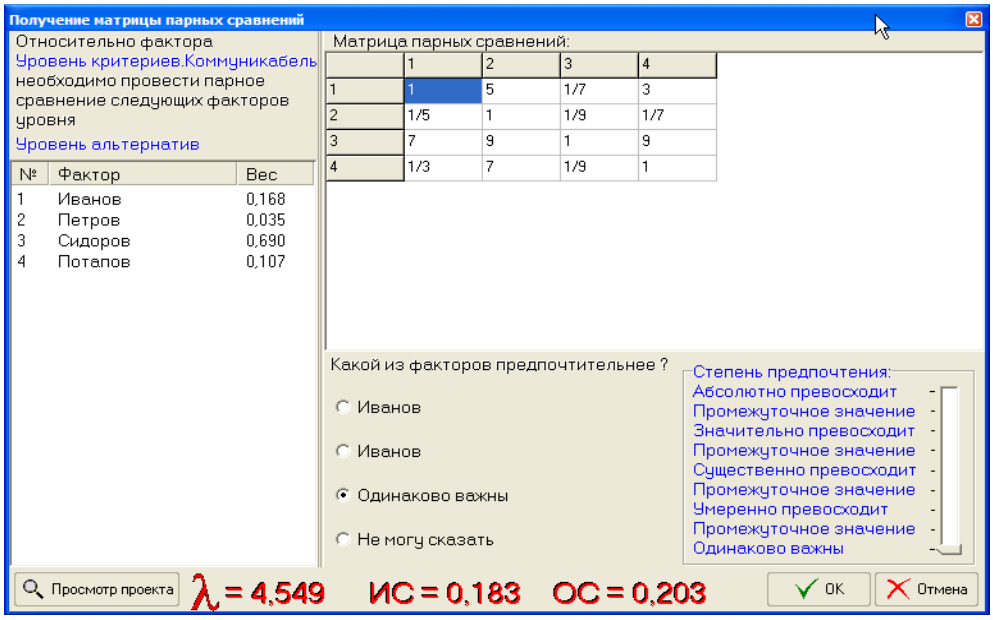


Рисунок 8 – сравнение по критерию коммуникабельности

Здесь мы видим очень интересную картину, по нашим наблюдениям наиболее коммуникабельным признан Сидоров, который по предыдущим критериям был аутсайдером, а вот Петров оказался позади всех, и признан замкнутым человеком.

Комментирую эту матрицу мы должны вспомнить постановку задачи, нам нужен заместитель начальника отдела информатизации, т.е. это должен быть человек хорошо разбирающийся в работе отдела, а значит техник по образованию, но с другой стороны это руководящая должность и возможно человек долго проработавший в отделе хоть и с гуманитарным образованием должен иметь равные шансы. Поэтому как мы видим на рис. 9, Иванову и Петрову проставлены одинаковые оценки, меньше всего у Сидорова т.к. руководитель должен иметь высшее образование.

Полученные матрицы парных сравнений по критерию образование (рис. 9):

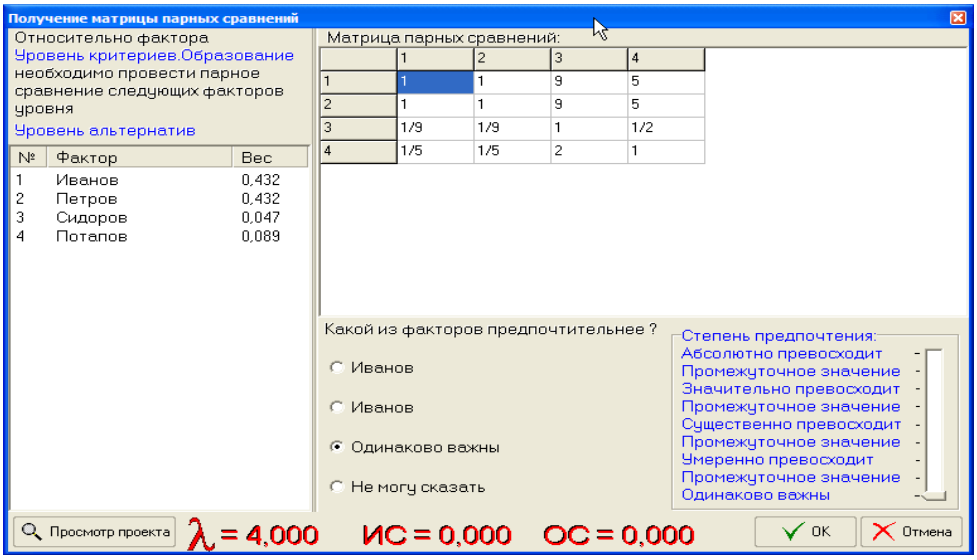


Рисунок 9 – Сравнение по критерию образования

Что для нас является важнее? Стаж, образование, коммуникабельность или ответственность? Думаю, уровняем шансы, нам нужен кандидат в равной степени удовлетворяющий всем параметрам. Расставляем приоритеты поровну. (рис. 10)

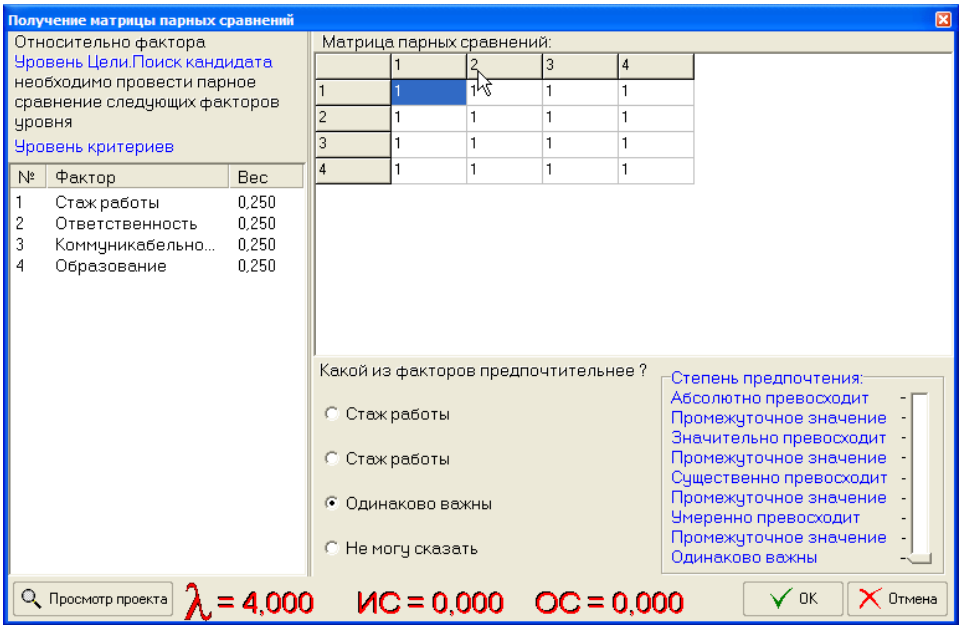


Рисунок 10 – Общие критерий кандидатов

Вычисления закончены, получаем результат (рис. 11):

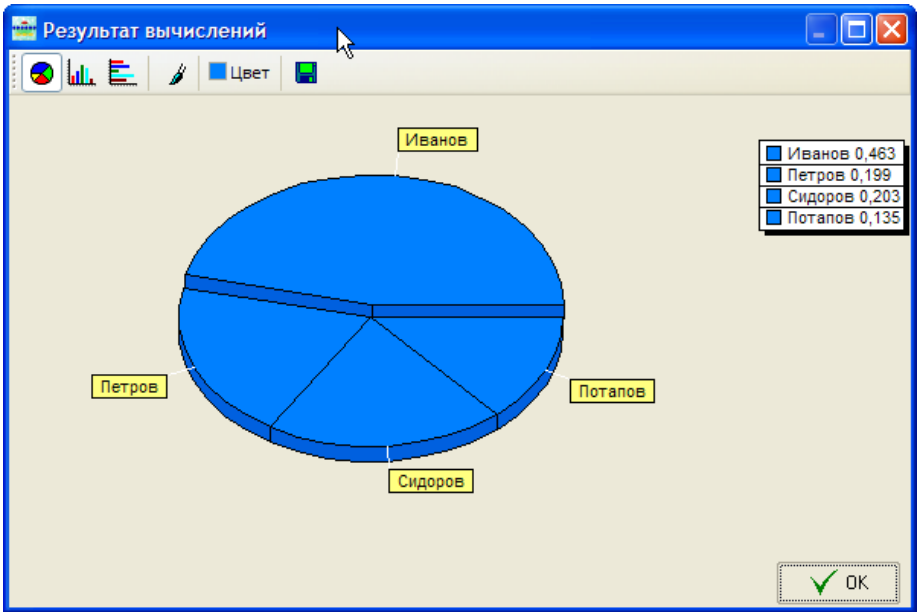


Рисунок 11-Получение результатов

Очевидно, что наиболее подходящим кандидатом на должность заместителя начальника отдела является Иванов.

Задание 1

С помощью программы «Выбор» решить следующую задачу. Нам необходимо произвести отбор кандидатов на освободившуюся должность старосты из числа студентов вашей группы. Кандидатов оценивать по нескольким критериям:

– успеваемость,

– ответственность,

– образование,

– коммуникабельность.

Произвести нужные вычисления и оформить выводы с предварительно сделанным проектом.

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6.1.1. Основная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л1.1	Петров, А. Е.	Математические модели принятия решений: учебно-методическое пособие	Москва: Издательский Дом МИСиС, 2018	http://www.iprbookshop.ru/78572.html
Л1.2	Муромцев, Д. Ю., Шамкин, В. Н.	Методы оптимизации и принятие проектных решений: учебное пособие для магистрантов по направлению 11.04.03	Тамбов: Тамбовский государственный технический университет, ЭБС АСВ, 2015	http://www.iprbookshop.ru/63866.html
Л1.3	Горелик, В. А.	Теория принятия решений: учебное пособие для магистрантов	Москва: Московский педагогический государственный университет, 2016	http://www.iprbookshop.ru/72518.html
6.1.2. Дополнительная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л2.1	Бережная, О. В., Бережная, Е. В.	Методы принятия управленческих решений: учебное пособие	Ставрополь: Северо-Кавказский федеральный университет, 2015	http://www.iprbookshop.ru/62960.html
	Авторы,	Заглавие	Издательство, год	Адрес
Л2.2	Казанская, О. В., Юн, С. Г., Альсова, О. К.	Модели и методы оптимизации. Практикум: учебное пособие	Новосибирск: Новосибирский государственный технический университет, 2012	http://www.iprbookshop.ru/45397.html
6.1.3. Методические разработки				
	Авторы,	Заглавие	Издательство, год	Адрес
Л3.1	Палинчак, Н. Ф., Ярославцева, В. Я.	Системный анализ, оптимизация и принятие решений: методические указания и задания для самостоятельной работы	Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2014	http://www.iprbookshop.ru/55156.html
Л3.2	Артюхин Г. А.	Теория систем и системный анализ. Практикум принятия решений: Учебное пособие	Казань: Казанский государственный архитектурно-строительный университет, ЭБС АСВ, 2016	http://www.iprbookshop.ru/73321.html

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Нейронные сети»
для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Нейронные сети» содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины - формирование у обучаемых знаний в области теоретических основ информационной безопасности и навыков практического обеспечения защиты информации и безопасного использования программных средств в вычислительных системах используемых на предприятиях.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

ОПК-4.1: Использует новые научные принципы исследований

ОПК-8.2: Планирует работу по разработке программных средств и проектов, составляет техническую документацию.

Изучив данный курс, студент должен:

Знать:

основные принципы организации информационных процессов в нейрокомпьютерных системах; основные архитектуры нейрокомпьютерных систем и области их применения; основные способы и правила обучения нейрокомпьютерных систем.

Уметь:

использовать навыки разработки и реализации программных моделей нейрокомпьютерных систем; делать оценки и сравнивать качество обучения и функционирования различных моделей нейрокомпьютерных систем.

Владеть:

современными достижениями в области разработки и коммерческом использовании нейрокомпьютерных систем и нейрокомпьютеров.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическое занятие 1 Установка операционной системы Windows XP. Создание учетных записей пользователя. Настройка локальной сети.

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание.1 Установите Windows 2003 server

1. В настройках BIOS установите следующую последовательность загрузки устройств: CD-ROM Жесткий диск. Эта настройка всегда зависит от типа BIOS, поэтому ее нельзя описать универсально. Подробную информацию вы найдете в описании, прилагающемся к вашей материнской плате.

2. В привод CD-ROM вставьте установочный компакт-диск с операционной системой Windows Server 2003 и перезагрузите компьютер.

3. Установка системы должна начаться автоматически. Если этого не происходит,

проверьте еще раз порядок загрузки в BIOS. Если же в компьютере уже была установлена какая-то операционная система, может случиться так, что для начала установки системе будет требоваться нажатие любой клавиши.

4. Включится текстовый режим установки и появится окно с надписью Windows Server 2003 Setup (Установка операционной системы Windows).

5. Ознакомьтесь с информацией программы установки и нажмите Enter.

6. Ознакомьтесь с информацией программы установки и нажмите Enter.

7. Ознакомьтесь с лицензионным соглашением и согласитесь с ним (клавиша F8).

8. Создайте раздел для ОС на всем жестком диске клавишей ENTER.

9. Выполните форматирование созданного раздела в файловой системе NTFS - нажмите ENTER. Дождитесь окончания форматирования раздела, и копирования файлов установки на него. В процессе копирования компьютер перезагрузится и продолжит установку автоматически.

10. Самостоятельно укажите параметры языка и раскладки клавиатуры и перейдите к следующему шагу кнопкой Далее.

11. Укажите регистрационные данные: ведите в поле Имя – USER о ведите в поле Организация – SIBCOL завершите ввод кнопкой Далее.

12. Введите в поле Ключ продукта лицензионный ключ и щелкните Далее.

13.Укажите вариант лицензирования при котором для каждого подключения требуется отдельная лицензия: о установите радиокнопку На сервере; введите в текстовое поле количество одновременных подключений, например 10; подтвердите параметры кнопкой Далее.

14. Укажите имя компьютера и пароль администратора:

Введите в поле Имя компьютера – WIN2003;

Введите в поле Пароль администратора – 123456;

Введите в поле Подтверждение - 123456.

Подтвердите сделанные изменения кнопкой Далее. Появится диалоговое окно сообщающее о том что пароль слишком простой.

Ознакомьтесь с информацией о том что вы указали простой пароль и продолжите установку кнопкой Да.

15. Укажите дату и время и щелкните Далее.

16. Установите сетевые параметры для использования статического IPадреса: о выберите радиокнопку Обычные параметры и щелкните Далее;

17. Укажите сетевую группу, например Workgroup и щелкните Далее.

18. Дождитесь окончания выполнения установки ОС. По окончании установки компьютер перезагрузится. После этого загрузится операционная система Windows 2003 Server.

Задание.2 Настройка локальной сети

Работа в рабочей группе

1. Щелкните правой кнопкой мыши на значке Мой компьютер, расположенном на Рабочем столе Windows, выберите в появившемся меню пункт Свойства

2. Перейдите ко вкладке Имя компьютера

3. Щелкните мышью на кнопке Изменить

4. Компьютер входит в сетевую рабочую группу, выберите режим Рабочей группы и наберите ее название в расположенном рядом поле.

5. Создать папку и ограничьте доступ следующим образом:

ПК 1 имеет доступ к ПК 3,4,6 на чтение и запись, к ПК7 на чтение, к остальным доступа не имеет.

ПК 2 имеет доступ к ПК 5,8 на чтение и запись, к ПК5 на чтение, к остальным доступа не имеет.

ПК 3 имеет доступ к ПК 7,9 на чтение и запись, к ПК4 на чтение, к остальным доступа не имеет.

ПК 4 имеет доступ к ПК 1,2 на чтение и запись, к ПК3 на чтение, к остальным доступа не имеет.

ПК 5 имеет доступ к ПК 4,7 на чтение и запись, к ПК2 на чтение, к остальным доступа не имеет.

ПК 6 имеет доступ к ПК 5,9 на чтение и запись, к ПК6 на чтение, к остальным доступа не имеет.

ПК 7 имеет доступ к ПК 6,8 на чтение и запись, к ПК8 на чтение, к остальным доступа не имеет.

ПК 8 имеет доступ к ПК 7,3 на чтение и запись, к ПК9 на чтение, к остальным доступа не имеет.

ПК 9 имеет доступ к ПК 2,6 на чтение и запись, к ПК10 на чтение, к остальным доступа не имеет.

ПК 10 имеет доступ к ПК 4,6 на чтение и запись, к ПК1 на чтение, к остальным доступа не имеет.

6.Заблокировать настройки рабочего стола.

7.Заблокировать сетевые настройки.

8.Создать папку на рабочем столе и сделать к ней общий доступ для всех на чтение.

Контрольные вопросы

1. Охарактеризуйте место операционной системы в программном обеспечении компьютеров, компьютерных систем и сетей.
2. В чем заключается основное назначение операционной системы?
3. Перечислите основные функции операционной системы.
4. Дайте понятие компьютерных ресурсов.
5. Дайте определение архитектуры операционных систем.
6. Перечислите поколения операционных систем.
7. Перечислите классификационные признаки операционной системы.
8. Охарактеризуйте виды интерфейсов операционных систем.
9. Опишите особенности эволюционных этапов операционных систем.
10. В чем заключается эффективность операционной системы?

Практическое занятие 2 Основные группы пользователей, идентификаторы безопасности (SID)

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание.

2.1. Ознакомьтесь с теоретическими основами защиты информации в ОС семейства Windows в настоящих указаниях и конспектах лекций.

2.2. Выполните задания 2.2.1-2.2.8 2.2.1.

2.2.1 При выполнении практического задания запустите в программе Oracle VM Virtualbox виртуальную машину Win7Test. Войдите в систему под учетной записью администратора. Все действия в пп 2.2.1-2.2.8 выполняйте в системе, работающей на виртуальной машине.

2.2.2. Создайте учетную запись нового пользователя testUser в оснастке «Управление компьютером» (compmgmt.msc). При создании новой учетной 22 записи запретите пользователю смену пароля и снимите ограничение на срок действия его пароля. Создайте новую группу ”testGroup” и включите в нее нового пользователя.

Удалите пользователя из других групп. Создайте на диске C: папку forTesting. Создайте или скопируйте в эту папку несколько текстовых файлов (*.txt).

2.2.3. С помощью команды runas запустите сеанс командной строки (cmd.exe) от имени вновь созданного пользователя. Командой whoami посмотрите SID пользователя и всех его групп, а также текущие привилегии пользователя. Строку запуска и результат работы этой и всех следующих консольных команд копируйте в файл протокола лабораторной работы.

2.2.4. Убедитесь в соответствии имени пользователя и полученного SID в реестре Windows. Найдите в реестре, какому пользователю в системе присвоен SID S-1-5-21-1957994488-492894223-170857768-1004 (Используйте ключ реестра HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList).

2.2.5. Командой whoami определите перечень текущих привилегий пользователя testUser. В сеансе командной строки пользователя попробуйте изменить системное время командой time. Чтобы предоставить пользователю подобную привилегию, запустите оснастку «Локальные параметры безопасности» (secpol.msc). Добавьте пользователя в список параметров политики «Изменение системного времени» раздела Локальные политики -> Назначение прав пользователя. После этого перезапустите сеанс командной строки от имени пользователя, убедитесь, что в списке привилегий добавилась SeSystemtimePriviege. Попробуйте изменить системное время командой time. Убедитесь, что привилегия «Завершение работы системы» (SeShutdownPrivilege) предоставлена пользователю testUser . После этого попробуйте завершить работу системы из сеанса командной строки пользователя командой shutdown –s. Добавть ему привелегию «Принудительное удаленное завершение» (SeRemoteShutdownPrivilege). Попробуйте завершить работу консольной командой еще раз (отменить команду завершения до ее непосредственного выполнения можно командой shutdown –a).

2.2.6. Ознакомьтесь с справкой по консольной команде icacls. Используя эту команду, просмотрите разрешения на папку c:\forTesting. Объясните все обозначения в описаниях прав пользователей и групп в выдаче команды. а) Разрешите пользователю testUser запись в папку forTesting, но запретите запись для группы testGroup. Попробуйте записать файлы или папки в forTesting от имени пользователя testUser. Объясните результат. Посмотрите эффективные разрешения пользователя testUser к папке forTesting в окне свойств папки. б) Используя стандартное окно свойств папки, задайте для пользователя testUser такие права доступа к папке, чтобы он мог записывать информацию в папку forTesting, но не мог просматривать ее содержимое. Проверьте, что папка forTesting является теперь для пользователя testUser “слепой”, запустив, например, от его имени файловый менеджер и попробовав записать файлы в папку, просмотреть ее содержимое, удалить файл из папки. в) Для вложенной папки forTesting\Docs отмените наследование ACL от родителя и разрешите пользователю просмотр, чтение и запись в папку. Проверьте, что для пользователя папка forTesting\Docs перестала быть “слепой” (например, 23 сделайте ее текущей в сеансе работы файлового менеджера от имени пользователя и создайте в ней новый файл). г) Снимите запрет на чтение папки forTesting для пользователя testUser. Используя команду icacls запретите этому пользователю доступ к файлам с расширением txt в папке forTesting. Убедитесь в недоступности файлов для пользователя. д) Командой icacls запретите пользователю все права на доступ к папке forTesting и разреште полный доступ к вложенной папке forTesting\Docs. Убедитесь в доступности папки forTesting\Docs для пользователя. Удалите у пользователя testUser привилегию SeChangeNotifyPrivilege. Попробуйте получить доступ к папке forTesting\Docs. Объясните результат. е) Запустите файловый менеджер от имени пользователя testUser и создайте в нем папку newFolder на диске C. Для папки newFolder очистите весь список ACL командой cacls. Попробуйте теперь получить доступ к папке от имени администратора и от имени пользователя. Кто и как теперь может вернуть доступ к папке? Верните полный доступ к папке для всех пользователей. ж) Создайте в разделе HKLM\Software реестра раздел testKey. Запретите пользователю testUser создание новых

разделов в этом разделе реестра. Создайте для раздела HKLM\Software\testKey SACL, позволяющий протоколировать отказы при создании новых подразделов, а также успехи при перечислении подразделов и запросе значений (предварительно проверьте, что в локальной политике безопасности соответствующий тип аудита включен). Попробуйте от имени пользователя testUser запустить regedit.exe и создать раздел в HKLM\Software. Убедитесь, что записи аудита были размещены в журнале безопасности (eventvwr.msc). з) С использованием команды whoami проверьте уровень целостности для пользователя testUser и администратора (учетная запись ВПИ). Запустите какоенибудь приложение (калькулятор, блокнот) от имени testUser и администратора. С использованием утилиты ProcessExplorer (можно найти в папке c:\Utils на виртуальной машине) проверьте уровень целостности запущенных приложений. Объясните разницу. Верните пользователю testUser права на полный доступ к папке forTesting. От имени администратора создайте в папке forTesting текстовый файл someText.txt. Измените уровень целостности этого файла до высокого с использованием команды icacls. Запустите блокнот от имени пользователя testUser, откройте в нём файл someText.txt, измените содержимое файла и попробуйте сохранить изменения. Объясните причину отказа в доступе. Как можно предоставить пользователю testUset доступ к файлу.

2.2.7. Шифрование файлов и папок средствами EFS. а) От имени пользователя testUser зашифруйте какой-нибудь файл на диске. Убедитесь, что после этого был создан сертификат пользователя, запустив оснастку certmgr.msc от имени пользователя (раздел Личные). Просмотрите основные параметры сертификата открытого ключа пользователя testUser (срок действия, используемые алгоритмы). Установите доверие к этому сертификату в вашей системе. б) Создайте в папке forTesting новую папку Encrypt. В папке Encrypt создайте или скопируйте в нее текстовый файл. Зашифруйте папку Encrypt и все ее содержимое из меню свойств папки от имени администратора. Попробуйте про24 смотреть или скопировать какой-нибудь файл этой папки от имени пользователя testUser. Объясните результат. Скопируйте зашифрованный файл в незашифрованную папку (например, forTesting). Убедитесь что он остался зашифрованным. Добавьте пользователя testUser в список имеющих доступа к файлу пользователей в окне свойств шифрования файла. Повторите попытку получить доступ к файлу от имени пользователя testUser. в) Создайте учетную запись нового пользователя agentUser, сделайте его членом группы Администраторы. Определите для пользователя agentUser роль агента восстановления EFS. Создайте в папке forTesting новый текстовый файл с произвольным содержимым. Зашифруйте этот файл от имени пользователя testUser. Убедитесь в окне подробностей шифрования файла, что пользователь agentUser является агентом восстановления для данного файла. Попробуйте прочитать содержимое файла от имени администратора и от имени пользователя agentUser. Объясните результат. г) Зашифруйте все текстовые файлы папки forTesting с использованием консольной команды шифрования cipher от имени пользователя testUser (предварительно снимите запрет на доступ к этим файлам, установленный в задании 2.2.6г). д) Убедитесь, что при копировании зашифрованных файлов на том с файловой системой, не поддерживающей EFS (например, FAT32 на флеш-накопителе), содержимое файла дешифруется.

2.2.8. После демонстрации результатов работы преподавателю восстановите исходное состояние системы: удалите созданные папки и файлы, разделы реестра, удалите учетную запись созданного пользователя и его группы, снимите с пользователя agentUser роль агента восстановления.

Контрольные вопросы

1. К какому классу безопасности относится ОС Windows по различным критериям оценки.
2. Каким образом пользователи идентифицируются в ОС Windows.
3. Что такое списки DACL и SACL.

4. Перечислите, каким образом можно запустить процесс от имени другого пользователя.

5. Как происходит проверка прав доступа пользователя к ресурсам в ОС Windows.

6. Что такое маркер безопасности, и какова его роль в модели безопасности Windows.

7. Как с использованием команды icacls добавить права на запись для всех файлов заданной папки.

8. Что такое уровень целостности? Как он влияет на права доступа субъ-ектов к объектам ОС? Как можно узнать и задать уровень целостности для объек-тов и субъектов?

9. Какие события подлежат аудиту в ОС Windows.

10. Каким образом шифруются файлы в файловой системе EFS? Что такое FEK? DDF? DDR.

11. Какие алгоритмы шифрования используются в EFS.

Практическое занятие 3 Настройка и просмотр сведений о системе. Автозагрузка файлов

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание1. Настройка и просмотр сведений о системе

Чтобы запустить программу «Сведения о системе», нажмите кнопку Пуск и выберите команду Справка и поддержка. Нажмите кнопку Поддержка на панели инструментов, затем щелкните ссылку Расширенные сведения о системе в группе Средства и ссылки в левой части окна. В правой части окна щелкните ссылку Просмотр дополнительных сведений о системе.

Настройка системы.

Чтобы запустить программу «MSconfig.exe», нажмите кнопку Пуск и выберите команду Справка и поддержка. Нажмите кнопку Поддержка на панели инструментов, затем щелкните ссылку Настройка системы в группе Средства и ссылки в левой части окна. В правой части окна щелкните ссылку Запуск программы настройки системы

После загрузки появляется окно с шестью вкладками:

- Общие - позволяет управлять параметрами запуска системы.
- Config.sys - редактирование файла config.sys.
- Autoexec.bat - соответственно.
- System.ini.
- Win.ini.

Задание2. Автозагрузка файлов

Автозагрузка - здесь перечислены все программы, которые запускаются при загрузке системы.

Очень удобно то, что все собрано в одном месте. Не надо лазить по реестру и файлам, чтобы посмотреть, что загружается на компьютере. Можно отключить загрузку любой программы или выполнение строки одного из перечисленных файлов, не правя ничего вручную. При этом комментарии будут расставлены автоматически, а программы, запускаемые из реестра, например, из раздела "Run", будут перенесены в раздел "Run-" (в конце соответствующего раздела добавляется символ "-").

Специальный текстовый конфигурационный файл «BOOT.INI», который используется в процессе загрузки — один из важнейших системных файлов «Windows XP».

Этот файл должен находиться в корневом каталоге загрузочного диска. Перед тем как модифицировать файл измените его атрибуты, так чтобы он не был «Только для

чтения» (щёлкните правой кнопкой мыши по файлу и выберите в контекстном меню последний пункт — «Свойства» и скиньте соответствующий флажок, устанавливаемый по умолчанию при инсталляции ОС).

Раздел [boot loader] служит для задания параметров загрузки операционной системы.

Параметр «timeout = 30» (по умолчанию) определяет количество секунд, в течение которого пользователь может выбирать один из пунктов меню. При «timeout = 0» загрузочное меню не отображается. «При timeout = -1 » меню находится на экране неограниченное время.

Параметр «default =» определяет путь к загружаемой по умолчанию системе. В разделе [operation systems] находятся сведения об установленных операционных системах.

При использовании двух операционных систем, например, «Windows Me» и «Windows XP», содержимое файла будет выглядеть примерно так:

[boot loader]

timeout=30

default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS

[operating systems]

multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional RU" /noexecute=optin /fastdetect"

Здесь:

«multi(0)» — порядковый номер адаптера, с которого осуществляется загрузка. Всегда имеет значение «0»,

«disk(0)» — всегда равен «0» (для большинства BIOS),

«rdisk(X)» — определяет порядковый номер жесткого диска с которого осуществляется загрузка (от «0» до «3»),

«partition(Y)» — порядковый номер раздела жесткого диска, с которого загружается ОС. Нумерация начинается с «1». Не нумеруются расширенные разделы MS-DOS (тип «5») и разделы типа «0 » — неиспользуемые.

Способы автозагрузки и отключение списков автозагрузки:

Реестр - в реестре автозагрузка представлена в нескольких местах:

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRun] - программы, которые запускаются при входе в систему. Данный раздел отвечает за запуск программ для всех пользователей системы.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunOnce] - программы, которые запускаются только один раз при входе пользователя в систему. После этого ключи программ автоматически удаляются из данного раздела реестра. Данный раздел отвечает за запуск программ для всех пользователей системы.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunOnceEx] - программы, которые запускаются только один раз, когда загружается система. Этот раздел используется при инсталляции программ, например для запуска настроечных модулей.

После этого ключи программ автоматически удаляются из данного раздела реестра. Данный раздел отвечает за запуск программ для всех пользователей системы.

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRun]- программы, которые запускаются при входе текущего пользователя в систему

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRunOnce] - программы, которые запускаются только один раз при входе текущего пользователя в систему. После этого ключи программ автоматически удаляются из данного раздела реестра.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunServices] - программы, которые загружаются при старте системы до входа пользователя в Windows.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunServic

esOnce] - программы отсюда загружаются только один раз, когда загружается система.

Например, чтобы автоматически запускать Блокнот при входе текущего пользователя, открываем Редактор реестра (regedit.exe), переходим в раздел

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRun]

и добавляем следующий ключ:

"NOTEPAD.EXE"="C:WINDOWSSystem32notepad.exe"

Откройте оснастку "Групповая политика" (gpedit.msc), перейдите на вкладку "Конфигурация компьютера - Административные шаблоны - Система". В правой части оснастки перейдите на пункт "Запускать указанные программы при входе в систему". По умолчанию эта политика не задана, но вы можете добавить туда программу: включаем политику, нажимаем кнопку "Показать - Добавить", указываем путь к программе, при этом если запускаемая программа находится в папке ..WINDOWSSystem32 то можно указать только название программы, иначе придется указать полный путь к программе. При этом в системном реестре в разделе [HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionpolicies] создается подраздел ExplorerRun с ключами добавленных программ. Пример:

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionpoliciesExplorerRun]

"1"="notepad.exe"

"2"="iexplore.exe"

В итоге получаем запуск Блокнота и Internet Explorer для всех пользователей.

Аналогично задается автозапуск для текущих пользователей, в оснастке "Групповая политика" это путь "Конфигурация пользователя - Административные шаблоны - Система", а в реестре раздел

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionPoliciesExplorerRun]

При этом программы из этого списка не отображаются в списке программ, доступных для отключения в msconfig.exe, а также определяются не всеми менеджерами автозагрузки.

6. Папка "Автозагрузка"- это папка, в которой хранятся ярлыки для программ запускаемых после входа пользователя в систему. Ярлыки в эту папку могут добавляться программами при их установке или пользователем самостоятельно. Существует две папки - общая для всех пользователей и индивидуальная для текущего пользователя. По умолчанию эти папки находятся здесь:

..Documents and SettingsAll UsersГлавное менюПрограммы Автозагрузка - это папка, программы из которой будут запускаться для всех пользователей компьютера.

..Documents and SettingsUsernameГлавное менюПрограммыАвтозагрузка- это папка, программы из которой будут запускаться для текущего пользователя (здесь он назван Username).

Посмотреть, какие программы у вас запускаются таким способом, можно, открыв меню "Пуск - Все программы - Автозагрузка". Если вы создадите в этой папке ярлык для какой-нибудь программы, она будет запускаться автоматически после входа пользователя в систему. Если при входе пользователя в систему удерживать нажатой клавишу "Shift", то программы из папок "Автозагрузка" запускаться не будут.

7. Смена папки автозагрузки- Windows считывает данные о пути к папке "Автозагрузка" из реестра. Этот путь прописан в следующих разделах:

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionExplorerUser Shell Folders]

"Common Startup"="%ALLUSERSPROFILE%Главное менюПрограммыАвтозагрузка" - для всех пользователей системы.

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionExplorerUser Shell Folders]

"Startup"="%USERPROFILE%Главное менюПрограммыАвтозагрузка" - для

текущего пользователя.

Сменив путь к папке, мы получим автозагрузку всех программ из указанной папки. Например:

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionExplorerUser Shell Folders]

"Startup"="c:mystartup" - система загрузит все программы, ярлыки которых находятся в папке c:mystartup, при этом папка "Автозагрузка" все так же будет отображаться в меню "Пуск", а если у пользователя в ней ничего не было, то он и не заметит подмены.

Практическое занятие 4 Шифрование файлов и папок. Сертификаты безопасности

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Теоретические сведения.

Шифрованная файловая система (EFS) позволяет безопасно хранить данные. EFS делает это возможным, благодаря шифрованию данных в выбранных файлах и папках NTFS. После того как файл или папка зашифрованы, с ними работают так же, как и с другими файлами или папками.

Шифрование является прозрачным для пользователя, зашифровавшего файл. Это означает, что перед использованием файл не нужно расшифровывать. Можно, как обычно, открыть файл и изменить его.

Использование EFS сходно с использованием разрешений для файлов и папок. Оба метода используются для ограничения доступа к данным. Но злоумышленник, получивший несанкционированный физический доступ к зашифрованным файлам и папкам, не сможет их прочитать. При его попытке открыть или скопировать зашифрованный файл или папку появиться сообщение, что доступа нет. Разрешения для файлов и папок не защищают от несанкционированных физических атак.

Шифрование и расшифровывание файлов выполняется установкой свойств шифрования для папок и файлов, как устанавливаются и другие атрибуты, например «только чтение», «сжатый» или «скрытый». Если шифруется папка, все файлы и подпапки, созданные в зашифрованной папке, автоматически шифруются. Рекомендуется использовать шифрование на уровне папки.

Файлы и папки могут также быть зашифрованы или расшифрованы с помощью команды **cipher**.

Шифрование файлов происходит следующим образом:

- Каждый файл имеет уникальный *ключ шифрования файла*, который позже используется для расшифровки данных файла.
- Ключ шифрования файла сам по себе зашифрован — он защищен открытым ключом пользователя, соответствующим сертификату EFS.
- Ключ шифрования файла также защищен открытым ключом каждого дополнительного пользователя EFS, уполномоченного расшифровывать файлы, и ключом каждого агента восстановления.

Сертификат и закрытый ключ системы EFS могут выдать несколько источников, включая созданные автоматически сертификаты и сертификаты, выданные центрами сертификации корпорации Майкрософт или другими центрами сертификации.

Расшифровка файлов происходит следующим образом:

- Для расшифровки файла необходимо сначала расшифровать его ключ шифрования. Ключ шифрования файла расшифровывается, если закрытый ключ пользователя совпадает с открытым
- Не только пользователь может расшифровать ключ шифрования файла. Другие назначенные пользователи и агенты восстановления также могут расшифровать

файл, используя собственный закрытый ключ.

Закрытые ключи содержатся в защищенном хранилище ключей, а не в диспетчере учетных записей безопасности (SAM) или в отдельном каталоге.

При работе с зашифрованными файлами и папками следует учитывать следующие сведения и рекомендации:

- Могут быть зашифрованы только файлы и папки, находящиеся на томах NTFS. Т. к. протокол WebDAV работает с файловой системой NTFS, для шифрования файлов с помощью протокола WebDAV требуется система NTFS.
- Сжатые файлы и папки не могут быть зашифрованы. Если шифрование выполняется для сжатого файла или папки, файл или папка преобразуются к состоянию без сжатия.
- Зашифрованные файлы могут стать расшифрованными, если файл копируется или перемещается на том, не являющийся томом NTFS
- При перемещении незашифрованных файлов в зашифрованную папку они автоматически шифруются в новой папке. Однако обратная операция не приведет к автоматической расшифровке файлов. Файлы необходимо явно расшифровать.
- Не могут быть зашифрованы файлы с атрибутом «Системный» и файлы в структуре папок системный корневой каталог.
- Шифрование папки или файла не защищает их от удаления. Любой пользователь, имеющий права на удаление, может удалить зашифрованные папки или файлы. По этой причине рекомендуется использование EFS в комбинации с разрешениями системы NTFS.
- Могут быть зашифрованы или расшифрованы файлы и папки на удаленном компьютере, для которого разрешено удаленное шифрование. Однако если зашифрованный файл открывается по сети, передаваемые при этом по сети данные не будут зашифрованы. Другие протоколы, например SSL/TLS или IPSec, должны использоваться для шифрования данных, передаваемых по сети. Протокол WebDAV позволяет локально зашифровать файл и передать его в зашифрованном виде.

Задание. Шифрование файлов и папок.

Зашифровать файл или папку:

- Щелкните правой кнопкой мыши файл или папку, которые требуется зашифровать, и выберите из контекстного меню команду **Свойства**.
- На вкладке **Общие** нажмите кнопку **Дополнительно**.
- Установите флажок **Шифровать содержимое для защиты данных**.

Примечания:

- Когда шифруется отдельный файл, система запросит подтверждение необходимости зашифровать также и папку, содержащую этот файл. Если подтверждение получено, все файлы и подпапки, добавляемые в папку в будущем, будут зашифрованы при добавлении.
- Когда шифруется папка, система запросит подтверждение необходимости зашифровать также файлы и подпапки в данной папке. Если подтверждение получено, все файлы и подпапки, расположенные в папке, шифруются, так же как и все файлы и подпапки, которые будут добавлены в папку в будущем. Если выбрано шифрование только папки, все файлы и подпапки в данной папке остаются незашифрованными. Однако любые файлы и подпапки, добавляемые в папку в будущем, будут зашифрованы при добавлении.

Расшифровать файл или папку:

1. Правой кнопкой мыши щелкните зашифрованную папку или диск, затем выберите команду **Свойства**.
2. На вкладке **Общие** нажмите кнопку **Дополнительно**.
3. Снимите флажок **Шифровать содержимое для защиты данных**.

Примечания:

- Когда расшифровывается папка, система запросит подтверждение необходимости расшифровывать также файлы и подпапки в данной папке. Если выбрано расшифровывание только папки, зашифрованные файлы и папки в расшифрованной папке остаются зашифрованными. Однако новые файлы и папки, создаваемые в расшифрованной папке, не будут зашифровываться автоматически.

Получить право на шифрование и расшифровку файлов:

1. Щелкните правой кнопкой мыши зашифрованный файл, который нужно изменить, и выберите команду Свойства.
2. На вкладке Общие нажмите кнопку Дополнительно.
3. В диалоговом окне Дополнительные атрибуты и нажмите кнопку Подробнее
4. Чтобы разрешить пользователю изменить этот файл нажмите кнопку Добавить и выполните следующие действия:

- Для добавления пользователя, чей сертификат на этом компьютере, выберите сертификат и нажмите кнопку **ОК**.
- Для просмотра сертификата на данном компьютере перед добавлением его к файлу выберите сертификат и затем нажмите кнопку **Просмотр сертификата**
- Для добавления пользователя из Active Directory нажмите кнопку **Найти пользователя** и затем кнопку **ОК**.

Чтобы запретить пользователю изменять выберите имя пользователя и нажмите кнопку **Удалить**.

Примечания:

- Нельзя группам предоставлять право доступа к шифрованию файлов.
- У всех пользователей, имеющих право шифрования и расшифровки файлов, сертификат должен быть на компьютере.

3. **Команда для шифрования Cipher**

Отображение или изменение шифрование папок и файлов на томах NTFS. Использованная без параметров команда **cipher** отображает состояние шифрования текущей папки и всех файлов, находящихся в ней.

Синтаксис

cipher [{**/e**|**/d**}] [**/s:***каталог*] [**/a**] [**/i**] [**/f**] [**/q**] [**/h**] [**/k**] [**/u**[**/n**]] [*путь* [...]] | [**/r:***имя_файла_без_расширения*] | [**/w:***путь*]

Параметры

/e - Шифрует указанные папки. Папки помечаются таким образом, чтобы файлы, которые будут добавляться в папку позже, также шифровались.

/d - Расшифровывает указанные папки. Папки помечаются таким образом, чтобы файлы, которые будут добавляться в папку позже, также шифровались.

/s: *каталог* - Выполняет выбранную операцию над указанной папкой и всеми подпапками в ней.

/a - Выполняет операцию над файлами и каталогами.

/i - Продолжение выполнения указанной операции даже после возникновения ошибок. По умолчанию выполнение **cipher** прекращается после возникновения ошибки.

/f - Выполнение шифрования или расшифровывания указанных объектов. По умолчанию уже зашифрованные или расшифрованные файлы пропускаются командой **cipher**.

/q - Включение в отчет только наиболее важных сведений.

/h - Отображение файлов с атрибутами «Скрытый» и «Системный». По умолчанию эти файлы не шифруются и не расшифровываются.

/k - Создание ключа шифрования файла для пользователя, выполнившего команду **cipher**. Если используется данный параметр, все остальные параметры команды **cipher** не учитываются.

/u - Обновление ключа шифрования файла пользователя или ключа агента восстановления на текущие ключи во всех зашифрованных файлах на локальном диске

(если эти ключи были изменены). Этот параметр используется только вместе с параметром **/n**.

/n - Запрещение обновления ключей. Данный параметр служит для поиска всех зашифрованных файлов на локальных дисках. Этот параметр используется только вместе с параметром **/u**.

путь - Указывает шаблон, файл или папку.

/r:*имя_файла_без_расширения* - Создание нового сертификата агента восстановления и закрытого ключа с последующей их записью в файлах с именем, указанным в параметре *имя_файла_без_расширения*. Если используется данный параметр, все остальные параметры команды **cipher** не учитываются.

/w:*путь* - Удаление данных из неиспользуемых разделов тома. Параметр *путь* может указывать на любой каталог нужного тома. Если используется данный параметр, все остальные параметры команды **cipher** не учитываются.

/? - Отображение справки в командной строке.

Примеры:

Чтобы зашифровать подпапку May в папке MonthlyReports с помощью команды **cipher**, введите следующую команду:

cipher /e monthlyreports\may

Чтобы зашифровать папку MonthlyReports, подпапки с January по December и подпапки Manufacturing в подпапках месяцев, введите:

cipher /e /s:monthlyreports

Чтобы зашифровать только файл Marketing.xls в подпапке May, введите:

cipher /e /a monthlyreports\may\marketing.xls

Чтобы зашифровать файл Marketing.xls, файл Maintenance.doc и подпапку Manufacturing (расположенные в папке May), введите:

cipher /e /a monthlyreports\may\ma*

Чтобы определить, зашифрована ли папка May, введите:

cipher monthlyreports\may

Чтобы определить, какие файлы зашифрованы в папке May, введите:

cipher monthlyreports\may*

Теоретические сведения. Основные сведения о сертификатах

Сертификат открытого ключа, обычно называемый просто сертификатом, — это документ с цифровой подписью, связывающий значение открытого ключа с удостоверением пользователя, устройства или службы, которым принадлежит соответствующий закрытый ключ.

Сертификаты могут выдаваться для различных целей, таких, как проверка подлинности пользователя Интернета, проверка подлинности веб-сервера, защита электронной почты (протокол S/MIME), безопасность IP (IPSec), безопасность на уровне транзакций (TLS) и подписание кода. Кроме того, центры сертификации выдают сертификаты другим центрам сертификации для создания иерархии сертификации.

Сертификат выдается так называемому *субъекту* сертификата. Выдачу и подписание сертификата осуществляет центр сертификации.

Как правило, сертификаты содержат следующие сведения.

- Значение открытого ключа субъекта.
- Сведения об идентификации субъекта, такие, как имя и адрес электронной почты.
- Срок действия (время, в течение которого сертификат считается действительным).
- Сведения для идентификации поставщика.
- цифровая подпись поставщика, заверяющая действительность связи между общим ключом субъекта и сведениями для его идентификации.

Сертификат действителен только в течение указанного в нем периода; каждый сертификат содержит даты *начала* и *окончания* срока действия. По окончании срока действия сертификата субъект устаревающего сертификата должен запросить новый

сертификат.

Одно из основных преимуществ использования сертификатов состоит в устранении необходимости использования на узлах паролей для отдельных субъектов, для предоставления доступа которым необходимо выполнять проверку их подлинности. Вместо этого узел просто устанавливает доверительные отношения с поставщиком сертификата.

Хранилища сертификатов

Windows XP хранит сертификат локально на компьютере или устройстве, которые запросили его, или, в случае пользователя, на компьютере или устройстве, которые пользователь использовал для запроса сертификата. Это место на запоминающем устройстве называется хранилищем сертификатов. Хранилище сертификатов часто содержит многочисленные сертификаты, возможно, полученные от различных центров сертификации.

С помощью оснастки «Сертификаты» можно отобразить хранилище сертификатов для пользователя, компьютера или службы в соответствии с целью, для которой сертификаты были выданы, или по категориям логических хранилищ. Когда сертификаты отображаются в соответствии с их категориями хранилища, можно также выбрать отображение физических хранилищ, показывая иерархию хранилищ сертификатов. (Это рекомендуется делать только опытным пользователям.)

Если пользователь имеет соответствующие права, он может импортировать или экспортировать сертификаты из любой папки хранилища сертификатов.

Сертификаты могут быть отображены по назначению и по логическим хранилищам. Отображение сертификатов по логическим хранилищам является установкой оснастки «Сертификаты» по умолчанию.

Импортирование и экспортирование сертификатов

Оснастка «Сертификаты» предоставляет административные средства для экспорта и импорта сертификатов, включая их пути сертификации и закрытые ключи, если это необходимо.

Импортирование сертификата

Импорт сертификата может понадобиться для выполнения перечисленных ниже задач.

- Установка сертификата, который был отправлен в файле другим пользователем, компьютером или центром сертификации.
- Восстановление поврежденного или утерянного сертификата, заархивированного ранее.
- Установка сертификата и связанного с ним закрытого ключа с компьютера, на котором владелец сертификата его использовал ранее.

Когда сертификат импортируется, он копируется из файла, который использует стандартный формат хранения сертификата, в хранилище сертификатов для учетной записи пользователя или учетной записи компьютера.

Экспортирование сертификата

Экспорт сертификата может понадобиться для выполнения перечисленных ниже задач.

- Архивирование сертификата.
- Архивирование сертификата и связанного с ним закрытого ключа.
- Копирование сертификата для использования на другом компьютере.
- Удаление сертификата и его закрытого ключа с компьютера владельца сертификата для установки на другом компьютере.

Когда сертификат экспортируется, он копируется из хранилища сертификатов в файл, использующий стандартный формат хранения сертификатов.

Чтобы открыть оснастку «Сертификаты», нажмите кнопку **Пуск**, выберите команду В**ыполнить** и введите **mmc** затем нажмите кнопку **OK**. В меню **Консоль** выберите команду **Открыть**, далее в дереве выберите необходимую

консоль и нажмите кнопку **Открыть**. Затем в дереве консоли щелкните папку **Сертификаты**.

Контрольные вопросы

1.Что такое сертификат и для чего он необходим?

2. В чем суть механизма защиты шифрованием?

3. В чем идея прозрачного шифрования?

4. Что такое консоль *MMC* и какие элементы управления может содержать консоль?

5. Назначение системы EFS.

6. Что такое Центр сертификации?

7. Какую информацию содержат сертификаты?

8. Какие виды ЦС используются службами Windows?

9. Какие типы сертификатов используются в Интернете?

10. Что такое Хранилище сертификатов и как его можно просмотреть?

11. Какую информацию содержат папки хранилища сертификатов?

12. Зачем запрашивают сертификаты и как это сделать?

13. Как осуществляется импорт и экспорт сертификатов?

Практическое занятие 5 Восстановление паролей пользователя при помощи программы LCP 5.04. Дисковые квоты в Windows XP

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание. Получение хэшей паролей

Существует несколько путей получения хэшей паролей, зависящих от их местонахождения и имеющегося доступа. Хэши паролей могут быть получены следующими способами: из файла SAM или его резервной копии, непосредственно из реестра операционной системы локального или удаленного компьютера, из реестра или Active Directory локального или удаленного компьютера внедрением DLL, посредством перехвата аутентификационных пакетов в сети.

LCP 5.04 - Программа предназначена для подбора паролей. Основные возможности: импорт информации об учетных записях пользователей; создание дампа паролей (методом pwdump; pwdump2); подбор паролей с применением словаря; подбор паролей гибридом атаки по словарю и последовательного перебора (добавление символов справа или слева от слов словаря); подбор пароля последовательным перебором комбинаций.

Операционные системы Windows NT/2000/XP/2003 хранят пароли в зашифрованном виде, называемом хэшами паролей (hash (англ.) - смесь, мешанина). Пароли не могут быть получены непосредственно из хэшей. Восстановление паролей заключается в вычислении хэшей по возможным паролям и сравнении их с имеющимися хэшами паролей. Аудит паролей включает в себя проверку возможных путей получения информации об учетных записях пользователей, результатом восстановления паролей является их представление в явном виде с учетом регистра.

Главное окно программы

Главное окно программы содержит меню, панель инструментов, панель состояния, список учетных записей пользователей, строку состояния.

В меню Файл содержатся команды выполнения действий с файлами, в меню Вид - команды задания вида главного окна программы, в меню Импорт - команды выполнения различного типа импорта информации об учетных записях пользователей, в меню Сеанс - команды управления сеансом аудита и восстановления паролей, в меню Справка - команды предоставления справочной информации о программе.

Кнопки панели инструментов дублируют наиболее часто используемые команды меню.

На панели состояния выводятся следующие данные о ходе восстановления:

Атака по словарю

Отображается информация о текущем слове словаря, количестве перебранных слов, общем количестве слов и проценте выполненной работы при восстановлении паролей атакой по словарю. Для задания параметров атаки по словарю выберите в меню Сеанс команду Параметры и в диалоговом окне Параметры перейдите на вкладку Атака по словарю.

Гибридная атака

Отображается информация о текущем слове словаря, количестве перебранных слов, общем количестве слов, проценте выполненной работы, начальной комбинации и конечной комбинации при восстановлении паролей гибридом атаки по словарю и последовательного перебора. Для задания параметров гибридной атаки выберите в меню Сеанс команду Параметры и в диалоговом окне Параметры перейдите на вкладку Гибридная атака.

Атака последовательным перебором

Отображается информация о последней комбинации последовательного перебора, проценте выполненной работы по последовательному перебору, количестве оставшегося времени, скорости перебора, начальной комбинации и конечной комбинации при восстановлении паролей атакой последовательным перебором. Для задания параметров атаки последовательным перебором выберите в меню Сеанс команду Параметры и в диалоговом окне Параметры перейдите на вкладку Атака последовательным перебором.

В некоторых случаях гибридная атака и атака последовательным перебором производятся по двум частям. В этом случае после начальной и конечной комбинаций в круглых скобках дополнительно приводится информация о номере части перебора в виде "(<НомерЧасти>/2)".

Список учетных записей пользователей содержит информацию об учетных записях, восстановление паролей которых производится. Найденные пароли или его части отображаются в столбцах LM-пароль и NT-пароль. Неизвестная половина LM-пароля при известной другой половине будет отображена символами "???????". Если по LM- и NT-хэшам доступна информация о длине пароля, ставится отметка в столбце <8 или >14.

2. Дисковые квоты

Общие сведения о дисковых квотах.

Дисковые квоты отслеживают и контролируют использование места на диске для томов NTFS. Администраторы могут настроить Windows таким образом, чтобы:

- запрещать использование дискового пространства сверх указанного предела и регистрировать случаи превышения этого предела пользователями;
- регистрировать события превышения пользователями указанного порога предупреждения, то есть отметки, при прохождении которой пользователь приближается к заданному для него пределу использования дискового пространства.

При включении дисковых квот можно задать два параметра: предельную квоту диска и порог предупреждения дисковой квоты. Например, можно задать для пользователя дисковую квоту в 500 мегабайт (МБ) и порог предупреждения дисковой квоты в 450 МБ. В этом случае пользователь сможет хранить на соответствующем томе не более 500 МБ файлов. Систему дисковых квот можно настроить таким образом, чтобы при сохранении пользователем на томе более 450 МБ файлов создавалась запись о событии системы. Для управления квотами на томе необходимо входить в состав группы «Администраторы».

Можно разрешить пользователям превышать заданные квоты. Включение квот без ограничения использования дискового пространства полезно в случаях, когда не требуется запрещать пользователям доступ к тому, но требуется отслеживать использование дискового пространства отдельными пользователями. Также можно

включить или отключить режим регистрации событий превышения пользователями заданных для них квот или порогов предупреждения. Отслеживание использования тома всеми пользователями начинается автоматически с момента включения дисковых квот для тома.

Квоты можно включать на локальных томах, сетевых томах и съемных дисках с файловой системой NTFS. Кроме того, для сетевых томов должен быть предоставлен общий доступ к корневому каталогу тома, а съемные диски должны быть предоставлены для общего доступа. Нельзя использовать сжатие файлов для предотвращения превышения пользователями заданных квот, поскольку сжатые файлы отслеживаются по их несжатому размеру.

При расчете использования тома сжатыми папками Windows, наоборот, использует размер папок после сжатия. Например, если папку размером 500 МБ сжать до 300 МБ, Windows сопоставит с квотой 300 МБ.

Включить дисковые квоты.

1. Щелкните правой кнопкой значок тома, для которого требуется включить дисковые квоты, и выберите команду Свойства.
2. В диалоговом окне Свойства откройте вкладку Квота.
3. Установите флажок Включить управление квотами на вкладке Квота.
4. Выберите один или несколько из следующих параметров и нажмите кнопку ОК:

- Не выделять место на диске при превышении квоты

Пользователи, превысившие квоту, получают сообщение об ошибке Windows "Недостаточно места на диске" и не могут записывать дополнительные данные в том без предварительного удаления или перемещения некоторых существующих файлов с диска.

В отдельных приложениях предусмотрен особый порядок действий в данной ситуации. Ситуация воспринимается приложением как переполнение диска. Если снять данный флажок, пользователи не смогут превышать предельную квоту. Включение квот без ограничения использования дискового пространства используется в случаях, когда не требуется запрещать пользователям доступ к тому, но требуется отслеживать использование дискового пространства отдельными пользователями. Можно также включить или отключить режим регистрации событий превышения пользователями заданных для них квот или порогов предупреждения.

- Выделять на диске не более

Укажите объем дискового пространства, выделяемого новым пользователям тома, а также порог, по достижении которого в системный журнал будет записано событие. Администраторы могут просматривать эти события в окне просмотра событий. Можно использовать десятичные числа (например, 20,5). Для дискового пространства и порога предупреждения в раскрывающемся списке выберите соответствующие наименования величин (например, Кбайт, Мбайт, Гбайт и т.п.).

- Регистрация превышения квоты пользователем

Если квоты включены, при превышении пользователями заданной предельной квоты в системный журнал на локальном компьютере заносится событие. Администраторы могут просматривать эти события в окне просмотра событий, отбирая их по типу.

По умолчанию события квоты записываются в системный журнал на локальном компьютере каждый час. Интервал записи событий квоты в системный журнал на локальном компьютере можно изменить с помощью команды fsutil behavior.

- Регистрация превышения порога предупреждения

Если квоты включены, при превышении пользователями заданного порога предупреждения в системный журнал на локальном компьютере заносится событие. Администраторы могут просматривать эти события в окне просмотра событий, отбирая их по типу.

По умолчанию события квоты записываются в системный журнал на локальном компьютере каждый час. Интервал записи событий квоты в системный журнал на локальном компьютере можно изменить с помощью команды fsutil behavior.

Контрольные вопросы:

1. Что такое аутентификация и идентификация?
2. Для чего применяются эти механизмы?
3. Что можно настроить с помощью вкладки Локальные политики безопасности?

Практическое занятие 6 Групповая политика. Политика аудита

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Теоретические сведения

Групповая политика Параметры групповой политики определяют различные компоненты окружения пользовательского рабочего стола, которыми управляет системный администратор (например, программы, доступные пользователям; программы, отображающиеся на пользовательском рабочем столе, и параметры меню **Пуск**). Чтобы создать конфигурацию рабочего стола для определенной группы пользователей используется оснастка «Групповая политика». Указанные параметры групповой политики содержатся в объекте групповой политики, который в свою очередь связан с выбранными объектами Active Directory — сайтами, доменами или подразделениями.

Конфигурация пользователя

Папка «Конфигурация пользователя» оснастки Групповая политика используется для задания политик, применяемых к пользователям независимо от того, какой компьютер используется для входа в систему.

Обычно узел «Конфигурация пользователя» содержит подпапки «Конфигурация программ», «Конфигурация Windows» и «Административные шаблоны», но поскольку оснастка «Групповая политика» имеет расширения, которые можно добавлять и удалять, то точный набор подпапок может различаться.

Конфигурация компьютера

С помощью узла «Конфигурация компьютера» в Групповой политике администраторы могут устанавливать политики, применяемые к компьютерам, вне зависимости от того, кто работает на них.

Узел «Конфигурация компьютера» обычно содержит подузлы «Конфигурация программ», «Конфигурация Windows» и «Административные шаблоны». Однако можно удалять и добавлять расширения групповой политики, поэтому подузлы могут отличаться от описанных выше.

Чтобы обновить групповую политику немедленно

1. Нажмите кнопку **Пуск** и выберите команду **Выполнить**.
2. В поле **Открыть** введите **gpupdate** и нажмите кнопку **OK**.

2. Административные шаблоны:

В Windows включен ряд файлов .adm. Эти текстовые файлы, называемые административными шаблонами, содержат сведения о политике для элементов, расположенных в папке «Административные шаблоны» в дереве консоли оснастки Групповая политика.

Файлы .adm

Файл .adm состоит из иерархии категорий и подкатегорий, которые вместе определяют отображение параметров политики. Кроме того, в файле содержатся следующие сведения:

- размещение параметров реестра, соответствующих каждому параметру;
- величина параметров или ограничений, связанных с каждым параметром;
- значение по умолчанию для большинства параметров;

- объяснение функции каждого параметра;
- версии Windows, поддерживающие каждый параметр.

Узел групповой политики «Административные шаблоны» содержит все сведения о политике на основе реестра. Конфигурация пользователя сохраняется в разделе **HKEY_CURRENT_USER** (HKCU), а конфигурация компьютера — в разделе **HKEY_LOCAL_MACHINE** (HKLM). В обоих этих разделах данные реестра, относящиеся к групповой политике, содержатся в папке \Software\Policies или \Software\Microsoft\Windows\CurrentVersion\Policies. Следовательно, параметры групповой политики хранятся в реестре в четырех областях.

Чтобы добавить или удалить файл административного шаблона (.adm):

1. Откройте редактируемый объект групповой политики и в дереве консоли щелкните правой кнопкой папку **Административные шаблоны**.
2. Выберите команду **Добавление и удаление шаблонов**.
3. Для удаления шаблона в списке **Текущие шаблоны политики** выберите шаблон и нажмите кнопку **Удалить**.

Если необходимо добавить шаблон, нажмите кнопку **Добавить**. В диалоговом окне **Шаблоны политики** щелкните шаблоны, которую требуется добавить, и нажмите кнопку **Открыть**.

4. В диалоговом окне **Добавление и удаление шаблонов** нажмите кнопку **Закрыть**.
3. **Политика аудита:**

Перед внедрением аудита необходимо выбрать политику аудита. Политика аудита указывает категории событий для аудита, связанных с безопасностью. При первой установке Windows XP Professional все категории аудита выключены. Включая аудит различных категорий событий, можно создавать политику аудита, удовлетворяющую всем требованиям организации.

Для проведения аудита можно выбрать следующие категории событий.

- Аудит событий входа в систему
- Аудит управления учетными записями
- Аудит доступа к службе каталогов
- Аудит входа в систему
- Аудит доступа к объектам
- Аудит изменения политики
- Аудит использования привилегий
- Аудит отслеживания процессов
- Аудит системных событий

Задание.

1. Создать оснастку Групповая политика и сохранить ее на рабочий стол.
2. Запретить пользователям шифровать данные, используя EFS
3. Установить политику аудита на успех входа/выхода из системы; изменение политики
4. Запретить администраторам изменять системное время
5. Скрыть команду "Свойства" в контекстном меню объекта "Мой компьютер"
6. Установить блокировку учетной записи пользователя на 5 минут при троекратном неверном вводе пароля.
7. Установить доступ к компьютеру из сети только администраторам

Контрольные вопросы

1. Что представляют собой групповые политики?
2. Дя чего используются групповые политики?
3. На какие области разделена утилита Group Policies?
4. Групповые политики по умолчанию

5. Дополнения групповой политики в Windows
6. Средства управления групповой политикой
7. Управление пользователями и группами AD
8. Оснастка Active Directory Users and Computers (Пользователи и компьютеры Active Directory)

9. Какова роль аудита в обеспечении безопасности компьютерной системы?
10. Где и каким образом формируется информация о событиях аудита?
11. Какая информация может быть получена в результате аудита?
12. Какие типы аудита вы знаете и для чего предназначен каждый из них?
13. Каким образом активизируется политика аудита?
14. Каким образом политика аудита применяется для выбранных объектов и пользователей?
15. В каких случаях целесообразно учитывать *Успех,* а когда целесообразно фиксировать *Отказ?*
16. Как пользоваться журналами безопасности?
17. Какие учетные записи дают право на настройку аудита и проверку результатов аудита? Каким образом администратор может использовать информацию об аудите для повышения безопасности системы?

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6.1.1. Основная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л1.1	Башлы П. Н., Бабаш А. В., Баранова Е. К.	Информационная безопасность и защита информации: Учебное пособие	Москва: Евразийский открытый институт, 2012	http://www.iprbookshop.ru/10677.html
6.1.2. Дополнительная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л2.1	В.В. Горгорова, А.В. Чернов	Информационная безопасность: учебное пособие	, 2011	https://ntb.donstu.ru/content/informacionnaya-bezopasnost
	Авторы,	Заглавие	Издательство, год	Адрес
Л2.2	Прохорова О. В.	Информационная безопасность и защита информации: Учебник	Самара: Самарский государственный архитектурно-строительный университет, ЭБС АСВ, 2014	http://www.iprbookshop.ru/43183.html
6.1.3. Методические разработки				
	Авторы,	Заглавие	Издательство, год	Адрес

Л3.1	ДГТУ, Каф. "ВСИиБ"; сост. В.В. Галушка	Методические указания к лабораторным работам по дисциплине «Информационная безопасность телекоммуникационных систем»	Ростов н/Д.: ИЦ ДГТУ, 2018	https://ntb.donstu.ru/content/metodicheskie-ukazaniya-k-laboratornym-rabotam-po-discipline-informacionnaya-bezopasnost-telekommunikacionnyh-sistem

6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"	
Э1	Артемов А.В. Информационная безопасность [Электронный ресурс]: курс лекций/ Артемов А.В.— Электрон. текстовые данные.— Орел: Межрегиональная Академия безопасности и выживания (МАБИВ), 2014.— 256 c.http://www.iprbookshop.ru/33430.html
Э2	Башлы П.Н. Информационная безопасность и защита информации [Электронный ресурс]: учебное пособие/ Башлы П.Н., Бабаш А.В., Баранова Е.К.— Электрон. текстовые данные.— М.: Евразийский открытый институт, 2012.— 311 c.http://www.iprbookshop.ru/10677
Э3	Галатенко В.А. Основы информационной безопасности [Электронный ресурс]/ Галатенко В.А.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 266 c.http://www.iprbookshop.ru/52209
6.3.1 Перечень программного обеспечения	
6.3.1.1	ОС Windows ;
6.3.1.2	Kaspersky Endpoint Security ;
6.3.1.3	Microsoft Office 2007 Professional Plus
6.3.1.4	Borland Developer Studio 2006
6.3.2 Перечень информационных справочных систем	
6.3.2.1	1. ЭБС «Консультант студента. Электронная библиотека» http://www.studmedlib.ru/ru
6.3.2.2	2. Профессиональные справочные системы "Техэксперт" http://www.cntd.ru/

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Математические модели представления знаний»
для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Информационные системы и технологии» содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины - формирование у обучаемых знаний в области теоретических основ информационной безопасности и навыков практического обеспечения защиты информации и безопасного использования программных средств в вычислительных системах используемых на предприятиях.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

ПК-4: Обладает способностью планировать, разрабатывать и совершенствовать системы управления охраной труда;

ПК-4.1: Определение целей и задач (политики), процессов управления охраной труда и оценка эффективности системы управления охраной труда.

Изучив данный курс, студент должен:

Знать:

о типовых разработанных средствах защиты информации и возможностях их использования в реальных задачах создания и внедрения информационных систем;

основы информационной безопасности и защиты информации;

принципы криптографических преобразований;

типовые программно-аппаратные средства и системы защиты информации от несанкционированного доступа в компьютерную среду.

Уметь:

реализовывать мероприятия для обеспечения на предприятии (в организации) деятельности в области защиты информации;

проводить анализ степени защищенности информации и осуществлять повышение уровня защиты с учетом развития математического и программного обеспечения вычислительных систем;

разрабатывать средства и системы защиты информации.

Владеть:

разработками средств и систем защиты информации;

навыками анализа степени защищенности информации.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическое занятие 1 Установка операционной системы Windows XP. Создание учетных записей пользователя. Настройка локальной сети.

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание.1 Установите Windows 2003 server

1. В настройках BIOS установите следующую последовательность загрузки

устройств: CD-ROM Жесткий диск. Эта настройка всегда зависит от типа BIOS, поэтому ее нельзя описать универсально. Подробную информацию вы найдете в описании, прилагающемся к вашей материнской плате.

2. В привод CD-ROM вставьте установочный компакт-диск с операционной системой Windows Server 2003 и перезагрузите компьютер.

3. Установка системы должна начаться автоматически. Если этого не происходит, проверьте еще раз порядок загрузки в BIOS. Если же в компьютере уже была установлена какая-то операционная система, может случиться так, что для начала установки системе будет требоваться нажатие любой клавиши.

4. Включится текстовый режим установки и появится окно с надписью Windows Server 2003 Setup (Установка операционной системы Windows).

5. Ознакомьтесь с информацией программы установки и нажмите Enter.

6. Ознакомьтесь с информацией программы установки и нажмите Enter.

7. Ознакомьтесь с лицензионным соглашением и согласитесь с ним (клавиша F8).

8. Создайте раздел для ОС на всем жестком диске клавишей ENTER.

9. Выполните форматирование созданного раздела в файловой системе NTFS - нажмите ENTER. Дождитесь окончания форматирования раздела, и копирования файлов установки на него. В процессе копирования компьютер перезагрузится и продолжит установку автоматически.

10. Самостоятельно укажите параметры языка и раскладки клавиатуры и перейдите к следующему шагу кнопкой Далее.

11. Укажите регистрационные данные: ведите в поле Имя – USER о ведите в поле Организация – SIBCOL завершите ввод кнопкой Далее.

12. Введите в поле Ключ продукта лицензионный ключ и щелкните Далее.

13.Укажите вариант лицензирования при котором для каждого подключения требуется отдельная лицензия: о установите радиокнопку На сервере; введите в текстовое поле количество одновременных подключений, например 10; подтвердите параметры кнопкой Далее.

14. Укажите имя компьютера и пароль администратора:

Введите в поле Имя компьютера – WIN2003;

Введите в поле Пароль администратора – 123456;

Введите в поле Подтверждение - 123456.

Подтвердите сделанные изменения кнопкой Далее. Появится диалоговое окно сообщающее о том что пароль слишком простой.

Ознакомьтесь с информацией о том что вы указали простой пароль и продолжите установку кнопкой Да.

15. Укажите дату и время и щелкните Далее.

16. Установите сетевые параметры для использования статического IPадреса: о выберите радиокнопку Обычные параметры и щелкните Далее;

17. Укажите сетевую группу, например Workgroup и щелкните Далее.

18. Дождитесь окончания выполнения установки ОС. По окончании установки компьютер перезагрузится. После этого загрузится операционная система Windows 2003 Server.

Задание.2 Настройка локальной сети

Работа в рабочей группе

1. Щелкните правой кнопкой мыши на значке Мой компьютер, расположенном на Рабочем столе Windows, выберите в появившемся меню пункт Свойства

2. Перейдите ко вкладке Имя компьютера

3. Щелкните мышью на кнопке Изменить

4. Компьютер входит в сетевую рабочую группу, выберите режим Рабочей группы и наберите ее название в расположенном рядом поле.

5. Создать папку и ограничьте доступ следующим образом:

ПК 1 имеет доступ к ПК 3,4,6 на чтение и запись, к ПК7 на чтение, к остальным доступа не имеет.

ПК 2 имеет доступ к ПК 5,8 на чтение и запись, к ПК5 на чтение, к остальным доступа не имеет.

ПК 3 имеет доступ к ПК 7,9 на чтение и запись, к ПК4 на чтение, к остальным доступа не имеет.

ПК 4 имеет доступ к ПК 1,2 на чтение и запись, к ПК3 на чтение, к остальным доступа не имеет.

ПК 5 имеет доступ к ПК 4,7 на чтение и запись, к ПК2 на чтение, к остальным доступа не имеет.

ПК 6 имеет доступ к ПК 5,9 на чтение и запись, к ПК6 на чтение, к остальным доступа не имеет.

ПК 7 имеет доступ к ПК 6,8 на чтение и запись, к ПК8 на чтение, к остальным доступа не имеет.

ПК 8 имеет доступ к ПК 7,3 на чтение и запись, к ПК9 на чтение, к остальным доступа не имеет.

ПК 9 имеет доступ к ПК 2,6 на чтение и запись, к ПК10 на чтение, к остальным доступа не имеет.

ПК 10 имеет доступ к ПК 4,6 на чтение и запись, к ПК1 на чтение, к остальным доступа не имеет.

6.Заблокировать настройки рабочего стола.

7.Заблокировать сетевые настройки.

8.Создать папку на рабочем столе и сделать к ней общий доступ для всех на чтение.

Контрольные вопросы

1. Охарактеризуйте место операционной системы в программном обеспечении компьютеров, компьютерных систем и сетей.
2. В чем заключается основное назначение операционной системы?
3. Перечислите основные функции операционной системы.
4. Дайте понятие компьютерных ресурсов.
5. Дайте определение архитектуры операционных систем.
6. Перечислите поколения операционных систем.
7. Перечислите классификационные признаки операционной системы.
8. Охарактеризуйте виды интерфейсов операционных систем.
9. Опишите особенности эволюционных этапов операционных систем.
10. В чем заключается эффективность операционной системы?

Практическое занятие 2 Основные группы пользователей, идентификаторы безопасности (SID)

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание.

2.1. Ознакомьтесь с теоретическими основами защиты информации в ОС семейства Windows в настоящих указаниях и конспектах лекций.

2.2. Выполните задания 2.2.1-2.2.8 2.2.1.

2.2.1 При выполнении практического задания запустите в программе Oracle VM Virtualbox виртуальную машину Win7Test. Войдите в систему под учетной записью

администратора. Все действия в пп 2.2.1-2.2.8 выполняйте в системе, работающей на виртуальной машине.

2.2.2. Создайте учетную запись нового пользователя testUser в оснастке «Управление компьютером» (compmgmt.msc). При создании новой учетной 22 записи запретите пользователю смену пароля и снимите ограничение на срок действия его пароля. Создайте новую группу ”testGroup” и включите в нее нового пользователя. Удалите пользователя из других групп. Создайте на диске C: папку forTesting. Создайте или скопируйте в эту папку несколько текстовых файлов (*.txt).

2.2.3. С помощью команды runas запустите сеанс командной строки (cmd.exe) от имени вновь созданного пользователя. Командой whoami посмотрите SID пользователя и всех его групп, а также текущие привилегии пользователя. Строку запуска и результат работы этой и всех следующих консольных команд копируйте в файл протокола лабораторной работы.

2.2.4. Убедитесь в соответствии имени пользователя и полученного SID в реестре Windows. Найдите в реестре, какому пользователю в системе присвоен SID S-1-5-21-1957994488-492894223-170857768-1004 (Используйте ключ реестра HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList).

2.2.5. Командой whoami определите перечень текущих привилегий пользователя testUser. В сеансе командной строки пользователя попробуйте изменить системное время командой time. Чтобы предоставить пользователю подобную привилегию, запустите оснастку «Локальные параметры безопасности» (secpol.msc). Добавьте пользователя в список параметров политики «Изменение системного времени» раздела Локальные политики -> Назначение прав пользователя. После этого перезапустите сеанс командной строки от имени пользователя, убедитесь, что в списке привилегий добавилась SeSystemtimePriviege. Попробуйте изменить системное время командой time. Убедитесь, что привилегия «Завершение работы системы» (SeShutdownPrivilege) предоставлена пользователю testUser . После этого попробуйте завершить работу системы из сеанса командной строки пользователя командой shutdown –s. Добавть ему привелегию «Принудительное удаленное завершение» (SeRemoteShutdownPrivilege). Попробуйте завершить работу консольной командой еще раз (отменить команду завершения до ее непосредственного выполнения можно командой shutdown –a).

2.2.6. Ознакомьтесь с справкой по консольной команде icacls. Используя эту команду, просмотрите разрешения на папку c:\forTesting. Объясните все обозначения в описаниях прав пользователей и групп в выдаче команды. а) Разрешите пользователю testUser запись в папку forTesting, но запретите запись для группы testGroup. Попробуйте записать файлы или папки в forTesting от имени пользователя testUser. Объясните результат. Посмотрите эффективные разрешения пользователя testUser к папке forTesting в окне свойств папки. б) Используя стандартное окно свойств папки, задайте для пользователя testUser такие права доступа к папке, чтобы он мог записывать информацию в папку forTesting, но не мог просматривать ее содержимое. Проверьте, что папка forTesting является теперь для пользователя testUser “слепой”, запустив, например, от его имени файловый менеджер и попробовав записать файлы в папку, просмотреть ее содержимое, удалить файл из папки. в) Для вложенной папки forTesting\Docs отмените наследование ACL от родителя и разрешите пользователю просмотр, чтение и запись в папку. Проверьте, что для пользователя папка forTesting\Docs перестала быть “слепой” (например, 23 сделайте ее текущей в сеансе работы файлового менеджера от имени пользователя и создайте в ней новый файл). г) Снимите запрет на чтение папки forTesting для пользователя testUser. Используя команду icacls запретите этому пользователю доступ к файлам с расширением txt в папке forTesting. Убедитесь в недоступности файлов для пользователя. д) Командой icacls запретите пользователю все права на доступ к папке forTesting и разреште полный доступ к вложенной папке forTesting\Docs. Убедитесь в доступности папки forTesting\Docs для пользователя. Удалите у пользователя testUser привилегию SeChangeNotifyPrivilege. Попробуйте получить доступ к папке

forTesting\Docs. Объясните результат. е) Запустите файловый менеджер от имени пользователя testUser и создайте в нем папку newFolder на диске C. Для папки newFolder очистите весь список ACL командой cacls. Попробуйте теперь получить доступ к папке от имени администратора и от имени пользователя. Кто и как теперь может вернуть доступ к папке? Верните полный доступ к папке для всех пользователей. ж) Создайте в разделе HKLM\Software реестра раздел testKey. Запретите пользователю testUser создание новых разделов в этом разделе реестра. Создайте для раздела HKLM\Software\testKey SACL, позволяющий протоколировать отказы при создании новых подразделов, а также успехи при перечислении подразделов и запросе значений (предварительно проверьте, что в локальной политике безопасности соответствующий тип аудита включен). Попробуйте от имени пользователя testUser запустить regedit.exe и создать раздел в HKLM\Software. Убедитесь, что записи аудита были размещены в журнале безопасности (eventvwr.msc). з) С использованием команды whoami проверьте уровень целостности для пользователя testUser и администратора (учетная запись ВПИ). Запустите какоенибудь приложение (калькулятор, блокнот) от имени testUser и администратора. С использованием утилиты ProcessExplorer (можно найти в папке c:\Utils на виртуальной машине) проверьте уровень целостности запущенных приложений. Объясните разницу. Верните пользователю testUser права на полный доступ к папке forTesting. От имени администратора создайте в папке forTesting текстовый файл someText.txt. Измените уровень целостности этого файла до высокого с использованием команды icacls. Запустите блокнот от имени пользователя testUser, откройте в нём файл someText.txt, измените содержимое файла и попробуйте сохранить изменения. Объясните причину отказа в доступе. Как можно предоставить пользователю testUset доступ к файлу.

2.2.7. Шифрование файлов и папок средствами EFS. а) От имени пользователя testUser зашифруйте какой-нибудь файл на диске. Убедитесь, что после этого был создан сертификат пользователя, запустив оснастку certmgr.msc от имени пользователя (раздел Личные). Просмотрите основные параметры сертификата открытого ключа пользователя testUser (срок действия, используемые алгоритмы). Установите доверие к этому сертификату в вашей системе. б) Создайте в папке forTesting новую папку Encrypt. В папке Encrypt создайте или скопируйте в нее текстовый файл. Зашифруйте папку Encrypt и все ее содержимое из меню свойств папки от имени администратора. Попробуйте про24 смотреть или скопировать какой-нибудь файл этой папки от имени пользователя testUser. Объясните результат. Скопируйте зашифрованный файл в незашифрованную папку (например, forTesting). Убедитесь что он остался зашифрованным. Добавьте пользователя testUser в список имеющих доступа к файлу пользователей в окне свойств шифрования файла. Повторите попытку получить доступ к файлу от имени пользователя testUser. в) Создайте учетную запись нового пользователя agentUser, сделайте его членом группы Администраторы. Определите для пользователя agentUser роль агента восстановления EFS. Создайте в папке forTesting новый текстовый файл с произвольным содержимым. Зашифруйте этот файл от имени пользователя testUser. Убедитесь в окне подробностей шифрования файла, что пользователь agentUser является агентом восстановления для данного файла. Попробуйте прочитать содержимое файла от имени администратора и от имени пользователя agentUser. Объясните результат. г) Зашифруйте все текстовые файлы папки forTesting с использованием консольной команды шифрования cipher от имени пользователя testUser (предварительно снимите запрет на доступ к этим файлам, установленный в задании 2.2.6г). д) Убедитесь, что при копировании зашифрованных файлов на том с файловой системой, не поддерживающей EFS (например, FAT32 на флеш-накопителе), содержимое файла дешифруется.

2.2.8. После демонстрации результатов работы преподавателю восстановите исходное состояние системы: удалите созданные папки и файлы, разделы реестра, удалите учетную запись созданного пользователя и его группы, снимите с пользователя agentUser роль агента восстановления.

Контрольные вопросы

1. К какому классу безопасности относится ОС Windows по различным критериям оценки.
2. Каким образом пользователи идентифицируются в ОС Windows.
3. Что такое списки DACL и SACL.
4. Перечислите, каким образом можно запустить процесс от имени другого пользователя.
5. Как происходит проверка прав доступа пользователя к ресурсам в ОС Windows.
6. Что такое маркер безопасности, и какова его роль в модели безопасности Windows.
7. Как с использованием команды icacls добавить права на запись для всех файлов заданной папки.
8. Что такое уровень целостности? Как он влияет на права доступа субъ-ектов к объектам ОС? Как можно узнать и задать уровень целостности для объек-тов и субъектов?
9. Какие события подлежат аудиту в ОС Windows.
10. Каким образом шифруются файлы в файловой системе EFS? Что такое FEK? DDF? DDR.
11. Какие алгоритмы шифрования используются в EFS.

Практическое занятие 3 Настройка и просмотр сведений о системе. Автозагрузка файлов

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание1. Настройка и просмотр сведений о системе

Чтобы запустить программу «Сведения о системе», нажмите кнопку Пуск и выберите команду Справка и поддержка. Нажмите кнопку Поддержка на панели инструментов, затем щелкните ссылку Расширенные сведения о системе в группе Средства и ссылки в левой части окна. В правой части окна щелкните ссылку Просмотр дополнительных сведений о системе.

Настройка системы.

Чтобы запустить программу «MSconfig.exe», нажмите кнопку Пуск и выберите команду Справка и поддержка. Нажмите кнопку Поддержка на панели инструментов, затем щелкните ссылку Настройка системы в группе Средства и ссылки в левой части окна. В правой части окна щелкните ссылку Запуск программы настройки системы

После загрузки появляется окно с шестью вкладками:

- Общие - позволяет управлять параметрами запуска системы.
- Config.sys - редактирование файла config.sys.
- Autoexec.bat - соответственно.
- System.ini.
- Win.ini.

Задание2. Автозагрузка файлов

Автозагрузка - здесь перечислены все программы, которые запускаются при загрузке системы.

Очень удобно то, что все собрано в одном месте. Не надо лазить по реестру и файлам, чтобы посмотреть, что загружается на компьютере. Можно отключить загрузку любой программы или выполнение строки одного из перечисленных файлов, не правя ничего вручную. При этом комментарии будут расставлены автоматически, а программы, запускаемые из реестра, например, из раздела "Run", будут перенесены в раздел "Run-" (в конце соответствующего раздела добавляется символ "-").

Специальный текстовый конфигурационный файл «BOOT.INI», который используется в процессе загрузки — один из важнейших системных файлов «Windows XP».

Этот файл должен находиться в корневом каталоге загрузочного диска. Перед тем как модифицировать файл измените его атрибуты, так чтобы он не был «Только для чтения» (щёлкните правой кнопкой мыши по файлу и выберите в контекстном меню последний пункт — «Свойства» и скиньте соответствующий флажок, устанавливаемый по умолчанию при инсталляции ОС).

Раздел [boot loader] служит для задания параметров загрузки операционной системы.

Параметр «timeout = 30» (по умолчанию) определяет количество секунд, в течение которого пользователь может выбирать один из пунктов меню. При «timeout = 0» загрузочное меню не отображается. «При timeout = -1 » меню находится на экране неограниченное время.

Параметр «default =» определяет путь к загружаемой по умолчанию системе. В разделе [operation systems] находятся сведения об установленных операционных системах.

При использовании двух операционных систем, например, «Windows Me» и «Windows XP», содержимое файла будет выглядеть примерно так:

[boot loader]

timeout=30

default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS

[operating systems]

multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional RU" /noexecute=optin /fastdetect"

Здесь:

«multi(0)» — порядковый номер адаптера, с которого осуществляется загрузка. Всегда имеет значение «0»,

«disk(0)» — всегда равен «0» (для большинства BIOS),

«rdisk(X)» — определяет порядковый номер жесткого диска с которого осуществляется загрузка (от «0» до «3»),

«partition(Y)» — порядковый номер раздела жесткого диска, с которого загружается ОС. Нумерация начинается с «1». Не нумеруются расширенные разделы MS-DOS (тип «5») и разделы типа «0 » — неиспользуемые.

Способы автозагрузки и отключение списков автозагрузки:

Реестр - в реестре автозагрузка представлена в нескольких местах:

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRun] - программы, которые запускаются при входе в систему. Данный раздел отвечает за запуск программ для всех пользователей системы.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunOnce] - программы, которые запускаются только один раз при входе пользователя в систему. После этого ключи программ автоматически удаляются из данного раздела реестра. Данный раздел отвечает за запуск программ для всех пользователей системы.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunOnceEx] - программы, которые запускаются только один раз, когда загружается система. Этот раздел используется при инсталляции программ, например для запуска настроечных модулей.

После этого ключи программ автоматически удаляются из данного раздела реестра. Данный раздел отвечает за запуск программ для всех пользователей системы.

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRun]- программы, которые запускаются при входе текущего пользователя в систему

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRunOnce] - программы, которые запускаются только один раз при входе текущего пользователя в систему. После этого ключи программ автоматически удаляются из данного раздела

реестра.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunServices] - программы, которые загружаются при старте системы до входа пользователя в Windows.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunServicesOnce] - программы отсюда загружаются только один раз, когда загружается система.

Например, чтобы автоматически запускать Блокнот при входе текущего пользователя, открываем Редактор реестра (regedit.exe), переходим в раздел

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRun]

и добавляем следующий ключ:

"NOTEPAD.EXE"="C:WINDOWSSystem32notepad.exe"

Откройте оснастку "Групповая политика" (gpedit.msc), перейдите на вкладку "Конфигурация компьютера - Административные шаблоны - Система". В правой части оснастки перейдите на пункт "Запускать указанные программы при входе в систему". По умолчанию эта политика не задана, но вы можете добавить туда программу: включаем политику, нажимаем кнопку "Показать - Добавить", указываем путь к программе, при этом если запускаемая программа находится в папке ..WINDOWSSystem32 то можно указать только название программы, иначе придется указать полный путь к программе. При этом в системном реестре в разделе [HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionpolicies] создается подраздел ExplorerRun с ключами добавленных программ. Пример:

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionpoliciesExplorerRun]

"1"="notepad.exe"

"2"="iexplore.exe"

В итоге получаем запуск Блокнота и Internet Explorer для всех пользователей.

Аналогично задается автозапуск для текущих пользователей, в оснастке "Групповая политика" это путь "Конфигурация пользователя - Административные шаблоны - Система", а в реестре раздел

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionPoliciesExplorerRun]

При этом программы из этого списка не отображаются в списке программ, доступных для отключения в msconfig.exe, а также определяются не всеми менеджерами автозагрузки.

6. Папка "Автозагрузка"- это папка, в которой хранятся ярлыки для программ запускаемых после входа пользователя в систему. Ярлыки в эту папку могут добавляться программами при их установке или пользователем самостоятельно. Существует две папки - общая для всех пользователей и индивидуальная для текущего пользователя. По умолчанию эти папки находятся здесь:

..Documents and SettingsAll UsersГлавное менюПрограммы Автозагрузка - это папка, программы из которой будут запускаться для всех пользователей компьютера.

..Documents and SettingsUsernameГлавное менюПрограммыАвтозагрузка- это папка, программы из которой будут запускаться для текущего пользователя (здесь он назван Username).

Посмотреть, какие программы у вас запускаются таким способом, можно, открыв меню "Пуск - Все программы - Автозагрузка". Если вы создадите в этой папке ярлык для какой-нибудь программы, она будет запускаться автоматически после входа пользователя в систему. Если при входе пользователя в систему удерживать нажатой клавишу "Shift", то программы из папок "Автозагрузка" запускаться не будут.

7. Смена папки автозагрузки- Windows считывает данные о пути к папке "Автозагрузка" из реестра. Этот путь прописан в следующих разделах:

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionExplorerUser Shell Folders]

"Common Startup"="%ALLUSERSPROFILE%Главное менюПрограммыАвтозагрузка" - для всех пользователей системы.

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionExplorerUser Shell Folders]

"Startup"="%USERPROFILE%Главное менюПрограммыАвтозагрузка" - для текущего пользователя.

Сменив путь к папке, мы получим автозагрузку всех программ из указанной папки. Например:

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionExplorerUser Shell Folders]

"Startup"="c:mystartup" - система загрузит все программы, ярлыки которых находятся в папке c:mystartup, при этом папка "Автозагрузка" все так же будет отображаться в меню "Пуск", а если у пользователя в ней ничего не было, то он и не заметит подмены.

Практическое занятие 4 Шифрование файлов и папок. Сертификаты безопасности

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Теоретические сведения.

Шифрованная файловая система (EFS) позволяет безопасно хранить данные. EFS делает это возможным, благодаря шифрованию данных в выбранных файлах и папках NTFS. После того как файл или папка зашифрованы, с ними работают так же, как и с другими файлами или папками.

Шифрование является прозрачным для пользователя, зашифровавшего файл. Это означает, что перед использованием файл не нужно расшифровывать. Можно, как обычно, открыть файл и изменить его.

Использование EFS сходно с использованием разрешений для файлов и папок. Оба метода используются для ограничения доступа к данным. Но злоумышленник, получивший несанкционированный физический доступ к зашифрованным файлам и папкам, не сможет их прочитать. При его попытке открыть или скопировать зашифрованный файл или папку появиться сообщение, что доступа нет. Разрешения для файлов и папок не защищают от несанкционированных физических атак.

Шифрование и расшифровывание файлов выполняется установкой свойств шифрования для папок и файлов, как устанавливаются и другие атрибуты, например «только чтение», «сжатый» или «скрытый». Если шифруется папка, все файлы и подпапки, созданные в зашифрованной папке, автоматически шифруются. Рекомендуется использовать шифрование на уровне папки.

Файлы и папки могут также быть зашифрованы или расшифрованы с помощью команды **cipher**.

Шифрование файлов происходит следующим образом:

- Каждый файл имеет уникальный *ключ шифрования файла*, который позже используется для расшифровки данных файла.
- Ключ шифрования файла сам по себе зашифрован — он защищен открытым ключом пользователя, соответствующим сертификату EFS.
- Ключ шифрования файла также защищен открытым ключом каждого дополнительного пользователя EFS, уполномоченного расшифровывать файлы, и ключом каждого агента восстановления.

Сертификат и закрытый ключ системы EFS могут выдать несколько источников, включая созданные автоматически сертификаты и сертификаты, выданные центрами сертификации корпорации Майкрософт или другими центрами сертификации.

Расшифровка файлов происходит следующим образом:

- Для расшифровки файла необходимо сначала расшифровать его ключ шифрования. Ключ шифрования файла расшифровывается, если закрытый ключ пользователя совпадает с открытым
- Не только пользователь может расшифровать ключ шифрования файла. Другие назначенные пользователи и агенты восстановления также могут расшифровать файл, используя собственный закрытый ключ.

Закрытые ключи содержатся в защищенном хранилище ключей, а не в диспетчере учетных записей безопасности (SAM) или в отдельном каталоге.

При работе с зашифрованными файлами и папками следует учитывать следующие сведения и рекомендации:

- Могут быть зашифрованы только файлы и папки, находящиеся на томах NTFS. Т. к. протокол WebDAV работает с файловой системой NTFS, для шифрования файлов с помощью протокола WebDAV требуется система NTFS.
- Сжатые файлы и папки не могут быть зашифрованы. Если шифрование выполняется для сжатого файла или папки, файл или папка преобразуются к состоянию без сжатия.
- Зашифрованные файлы могут стать расшифрованными, если файл копируется или перемещается на том, не являющийся томом NTFS
- При перемещении незашифрованных файлов в зашифрованную папку они автоматически шифруются в новой папке. Однако обратная операция не приведет к автоматической расшифровке файлов. Файлы необходимо явно расшифровать.
- Не могут быть зашифрованы файлы с атрибутом «Системный» и файлы в структуре папок системный корневой каталог.
- Шифрование папки или файла не защищает их от удаления. Любой пользователь, имеющий права на удаление, может удалить зашифрованные папки или файлы. По этой причине рекомендуется использование EFS в комбинации с разрешениями системы NTFS.
- Могут быть зашифрованы или расшифрованы файлы и папки на удаленном компьютере, для которого разрешено удаленное шифрование. Однако если зашифрованный файл открывается по сети, передаваемые при этом по сети данные не будут зашифрованы. Другие протоколы, например SSL/TLS или IPSec, должны использоваться для шифрования данных, передаваемых по сети. Протокол WebDAV позволяет локально зашифровать файл и передать его в зашифрованном виде.

Задание. Шифрование файлов и папок.

Зашифровать файл или папку:

- Щелкните правой кнопкой мыши файл или папку, которые требуется зашифровать, и выберите из контекстного меню команду **Свойства**.
- На вкладке **Общие** нажмите кнопку **Дополнительно**.
- Установите флажок **Шифровать содержимое для защиты данных**.

Примечания:

- Когда шифруется отдельный файл, система запросит подтверждение необходимости зашифровать также и папку, содержащую этот файл. Если подтверждение получено, все файлы и подпапки, добавляемые в папку в будущем, будут зашифрованы при добавлении.
- Когда шифруется папка, система запросит подтверждение необходимости зашифровать также файлы и подпапки в данной папке. Если подтверждение получено, все файлы и подпапки, расположенные в папке, шифруются, так же как и все файлы и подпапки, которые будут добавлены в папку в будущем. Если выбрано шифрование только папки, все файлы и подпапки в данной папке остаются незашифрованными. Однако любые файлы и подпапки, добавляемые в папку в будущем, будут зашифрованы при добавлении.

Расшифровать файл или папку:

1. Правой кнопкой мыши щелкните зашифрованную папку или диск, затем выберите команду **Свойства**.
2. На вкладке **Общие** нажмите кнопку **Дополнительно**.
3. Снимите флажок **Шифровать содержимое для защиты данных**.

Примечания:

- Когда расшифровывается папка, система запросит подтверждение необходимости расшифровывать также файлы и подпапки в данной папке. Если выбрано расшифровывание только папки, зашифрованные файлы и папки в расшифрованной папке остаются зашифрованными. Однако новые файлы и папки, создаваемые в расшифрованной папке, не будут зашифровываться автоматически.

Получить право на шифрование и расшифровку файлов:

1. Щелкните правой кнопкой мыши зашифрованный файл, который нужно изменить, и выберите команду Свойства.
2. На вкладке Общие нажмите кнопку Дополнительно.
3. В диалоговом окне Дополнительные атрибуты и нажмите кнопку Подробнее
4. Чтобы разрешить пользователю изменить этот файл нажмите кнопку Добавить и выполните следующие действия:

- Для добавления пользователя, чей сертификат на этом компьютере, выберите сертификат и нажмите кнопку **ОК**.
- Для просмотра сертификата на данном компьютере перед добавлением его к файлу выберите сертификат и затем нажмите кнопку **Просмотр сертификата**
- Для добавления пользователя из Active Directory нажмите кнопку **Найти пользователя** и затем кнопку **ОК**.

Чтобы запретить пользователю изменять выберите имя пользователя и нажмите кнопку **Удалить**.

Примечания:

- Нельзя группам предоставлять право доступа к шифрованию файлов.
- У всех пользователей, имеющих право шифрования и расшифровки файлов, сертификат должен быть на компьютере.

3. **Команда для шифрования Cipher**

Отображение или изменение шифрование папок и файлов на томах NTFS. Использованная без параметров команда **cipher** отображает состояние шифрования текущей папки и всех файлов, находящихся в ней.

Синтаксис

cipher [{**/e**|**/d**}] [**/s:***каталог*] [**/a**] [**/i**] [**/f**] [**/q**] [**/h**] [**/k**] [**/u**[**/n**]] [*путь* [...]] | [**/r:***имя_файла_без_расширения*] | [**/w:***путь*]

Параметры

/e - Шифрует указанные папки. Папки помечаются таким образом, чтобы файлы, которые будут добавляться в папку позже, также шифровались.

/d - Расшифровывает указанные папки. Папки помечаются таким образом, чтобы файлы, которые будут добавляться в папку позже, также шифровались.

/s: *каталог* - Выполняет выбранную операцию над указанной папкой и всеми подпапками в ней.

/a - Выполняет операцию над файлами и каталогами.

/i - Продолжение выполнения указанной операции даже после возникновения ошибок. По умолчанию выполнение **cipher** прекращается после возникновения ошибки.

/f - Выполнение шифрования или расшифровывания указанных объектов. По умолчанию уже зашифрованные или расшифрованные файлы пропускаются командой **cipher**.

/q - Включение в отчет только наиболее важных сведений.

/h - Отображение файлов с атрибутами «Скрытый» и «Системный». По умолчанию эти файлы не шифруются и не расшифровываются.

/k - Создание ключа шифрования файла для пользователя, выполнившего команду **cipher**. Если используется данный параметр, все остальные параметры команды **cipher** не учитываются.

/u - Обновление ключа шифрования файла пользователя или ключа агента восстановления на текущие ключи во всех зашифрованных файлах на локальном диске (если эти ключи были изменены). Этот параметр используется только вместе с параметром **/n**.

/n - Запрещение обновления ключей. Данный параметр служит для поиска всех зашифрованных файлов на локальных дисках. Этот параметр используется только вместе с параметром **/u**.

путь - Указывает шаблон, файл или папку.

/r:*имя_файла_без_расширения* - Создание нового сертификата агента восстановления и закрытого ключа с последующей их записью в файлах с именем, указанным в параметре *имя_файла_без_расширения*. Если используется данный параметр, все остальные параметры команды **cipher** не учитываются.

/w:*путь* - Удаление данных из неиспользуемых разделов тома. Параметр *путь* может указывать на любой каталог нужного тома. Если используется данный параметр, все остальные параметры команды **cipher** не учитываются.

/? - Отображение справки в командной строке.

Примеры:

Чтобы зашифровать подпапку May в папке MonthlyReports с помощью команды **cipher**, введите следующую команду:

cipher /e monthlyreports\may

Чтобы зашифровать папку MonthlyReports, подпапки с January по December и подпапки Manufacturing в подпапках месяцев, введите:

cipher /e /s:monthlyreports

Чтобы зашифровать только файл Marketing.xls в подпапке May, введите:

cipher /e /a monthlyreports\may\marketing.xls

Чтобы зашифровать файл Marketing.xls, файл Maintenance.doc и подпапку Manufacturing (расположенные в папке May), введите:

cipher /e /a monthlyreports\may\ma*

Чтобы определить, зашифрована ли папка May, введите:

cipher monthlyreports\may

Чтобы определить, какие файлы зашифрованы в папке May, введите:

cipher monthlyreports\may*

Теоретические сведения. Основные сведения о сертификатах

Сертификат открытого ключа, обычно называемый просто сертификатом, — это документ с цифровой подписью, связывающий значение открытого ключа с удостоверением пользователя, устройства или службы, которым принадлежит соответствующий закрытый ключ.

Сертификаты могут выдаваться для различных целей, таких, как проверка подлинности пользователя Интернета, проверка подлинности веб-сервера, защита электронной почты (протокол S/MIME), безопасность IP (IPSec), безопасность на уровне транзакций (TLS) и подписание кода. Кроме того, центры сертификации выдают сертификаты другим центрам сертификации для создания иерархии сертификации.

Сертификат выдается так называемому *субъекту* сертификата. Выдачу и подписание сертификата осуществляет центр сертификации.

Как правило, сертификаты содержат следующие сведения.

- Значение открытого ключа субъекта.
- Сведения об идентификации субъекта, такие, как имя и адрес электронной почты.
- Срок действия (время, в течение которого сертификат считается действительным).
- Сведения для идентификации поставщика.

- цифровая подпись поставщика, заверяющая действительность связи между общим ключом субъекта и сведениями для его идентификации.

Сертификат действителен только в течение указанного в нем периода; каждый сертификат содержит даты *начала* и *окончания* срока действия. По окончании срока действия сертификата субъект устаревающего сертификата должен запросить новый сертификат.

Одно из основных преимуществ использования сертификатов состоит в устранении необходимости использования на узлах паролей для отдельных субъектов, для предоставления доступа которым необходимо выполнять проверку их подлинности. Вместо этого узел просто устанавливает доверительные отношения с поставщиком сертификата.

Хранилища сертификатов

Windows XP хранит сертификат локально на компьютере или устройстве, которые запросили его, или, в случае пользователя, на компьютере или устройстве, которые пользователь использовал для запроса сертификата. Это место на запоминающем устройстве называется хранилищем сертификатов. Хранилище сертификатов часто содержит многочисленные сертификаты, возможно, полученные от различных центров сертификации.

С помощью оснастки «Сертификаты» можно отобразить хранилище сертификатов для пользователя, компьютера или службы в соответствии с целью, для которой сертификаты были выданы, или по категориям логических хранилищ. Когда сертификаты отображаются в соответствии с их категориями хранилища, можно также выбрать отображение физических хранилищ, показывая иерархию хранилищ сертификатов. (Это рекомендуется делать только опытным пользователям.)

Если пользователь имеет соответствующие права, он может импортировать или экспортировать сертификаты из любой папки хранилища сертификатов.

Сертификаты могут быть отображены по назначению и по логическим хранилищам. Отображение сертификатов по логическим хранилищам является установкой оснастки «Сертификаты» по умолчанию.

Импортирование и экспортирование сертификатов

Оснастка «Сертификаты» предоставляет административные средства для экспорта и импорта сертификатов, включая их пути сертификации и закрытые ключи, если это необходимо.

Импортирование сертификата

Импорт сертификата может понадобиться для выполнения перечисленных ниже задач.

- Установка сертификата, который был отправлен в файле другим пользователем, компьютером или центром сертификации.
- Восстановление поврежденного или утерянного сертификата, заархивированного ранее.
- Установка сертификата и связанного с ним закрытого ключа с компьютера, на котором владелец сертификата его использовал ранее.

Когда сертификат импортируется, он копируется из файла, который использует стандартный формат хранения сертификата, в хранилище сертификатов для учетной записи пользователя или учетной записи компьютера.

Экспортирование сертификата

Экспорт сертификата может понадобиться для выполнения перечисленных ниже задач.

- Архивирование сертификата.
- Архивирование сертификата и связанного с ним закрытого ключа.
- Копирование сертификата для использования на другом компьютере.
- Удаление сертификата и его закрытого ключа с компьютера владельца сертификата для установки на другом компьютере.

Когда сертификат экспортируется, он копируется из хранилища сертификатов в файл, использующий стандартный формат хранения сертификатов.

Чтобы открыть оснастку «Сертификаты», нажмите кнопку **Пуск**, выберите команду В**ыполнить** и введите **mmc** затем нажмите кнопку **OK**. В меню **Консоль** выберите команду **Открыть**, далее в дереве выберите необходимую консоль и нажмите кнопку **Открыть**. Затем в дереве консоли щелкните папку **Сертификаты**.

Контрольные вопросы

1.Что такое сертификат и для чего он необходим?

2. В чем суть механизма защиты шифрованием?

3. В чем идея прозрачного шифрования?

4. Что такое консоль *MMC* и какие элементы управления может содержать консоль?

5. Назначение системы EFS.

6. Что такое Центр сертификации?

7. Какую информацию содержат сертификаты?

8. Какие виды ЦС используются службами Windows?

9. Какие типы сертификатов используются в Интернете?

10. Что такое Хранилище сертификатов и как его можно просмотреть?

11. Какую информацию содержат папки хранилища сертификатов?

12. Зачем запрашивают сертификаты и как это сделать?

13. Как осуществляется импорт и экспорт сертификатов?

Практическое занятие 5 Восстановление паролей пользователя при помощи программы LCP 5.04. Дисковые квоты в Windows XP

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание. Получение хэшей паролей

Существует несколько путей получения хэшей паролей, зависящих от их местонахождения и имеющегося доступа. Хэши паролей могут быть получены следующими способами: из файла SAM или его резервной копии, непосредственно из реестра операционной системы локального или удаленного компьютера, из реестра или Active Directory локального или удаленного компьютера внедрением DLL, посредством перехвата аутентификационных пакетов в сети.

LCP 5.04 - Программа предназначена для подбора паролей. Основные возможности: импорт информации об учетных записях пользователей; создание дампа паролей (методом pwdump; pwdump2); подбор паролей с применением словаря; подбор паролей гибридом атаки по словарю и последовательного перебора (добавление символов справа или слева от слов словаря); подбор пароля последовательным перебором комбинаций.

Операционные системы Windows NT/2000/XP/2003 хранят пароли в зашифрованном виде, называемом хэшами паролей (hash (англ.) - смесь, мешанина). Пароли не могут быть получены непосредственно из хэшей. Восстановление паролей заключается в вычислении хэшей по возможным паролям и сравнении их с имеющимися хэшами паролей. Аудит паролей включает в себя проверку возможных путей получения информации об учетных записях пользователей, результатом восстановления паролей является их представление в явном виде с учетом регистра.

Главное окно программы

Главное окно программы содержит меню, панель инструментов, панель состояния, список учетных записей пользователей, строку состояния.

В меню Файл содержатся команды выполнения действий с файлами, в меню Вид - команды задания вида главного окна программы, в меню Импорт - команды выполнения различного типа импорта информации об учетных записях пользователей, в меню Сеанс - команды управления сеансом аудита и восстановления паролей, в меню Справка - команды предоставления справочной информации о программе.

Кнопки панели инструментов дублируют наиболее часто используемые команды меню.

На панели состояния выводятся следующие данные о ходе восстановления:

Атака по словарю

Отображается информация о текущем слове словаря, количестве перебранных слов, общем количестве слов и проценте выполненной работы при восстановлении паролей атакой по словарю. Для задания параметров атаки по словарю выберите в меню Сеанс команду Параметры и в диалоговом окне Параметры перейдите на вкладку Атака по словарю.

Гибридная атака

Отображается информация о текущем слове словаря, количестве перебранных слов, общем количестве слов, проценте выполненной работы, начальной комбинации и конечной комбинации при восстановлении паролей гибридом атаки по словарю и последовательного перебора. Для задания параметров гибридной атаки выберите в меню Сеанс команду Параметры и в диалоговом окне Параметры перейдите на вкладку Гибридная атака.

Атака последовательным перебором

Отображается информация о последней комбинации последовательного перебора, проценте выполненной работы по последовательному перебору, количестве оставшегося времени, скорости перебора, начальной комбинации и конечной комбинации при восстановлении паролей атакой последовательным перебором. Для задания параметров атаки последовательным перебором выберите в меню Сеанс команду Параметры и в диалоговом окне Параметры перейдите на вкладку Атака последовательным перебором.

В некоторых случаях гибридная атака и атака последовательным перебором производятся по двум частям. В этом случае после начальной и конечной комбинаций в круглых скобках дополнительно приводится информация о номере части перебора в виде "(<НомерЧасти>/2)".

Список учетных записей пользователей содержит информацию об учетных записях, восстановление паролей которых производится. Найденные пароли или его части отображаются в столбцах LM-пароль и NT-пароль. Неизвестная половина LM-пароля при известной другой половине будет отображена символами "???????". Если по LM- и NT-хэшам доступна информация о длине пароля, ставится отметка в столбце <8 или >14.

2. Дисковые квоты

Общие сведения о дисковых квотах.

Дисковые квоты отслеживают и контролируют использование места на диске для томов NTFS. Администраторы могут настроить Windows таким образом, чтобы:

- запрещать использование дискового пространства сверх указанного предела и регистрировать случаи превышения этого предела пользователями;
- регистрировать события превышения пользователями указанного порога предупреждения, то есть отметки, при прохождении которой пользователь приближается к заданному для него пределу использования дискового пространства.

При включении дисковых квот можно задать два параметра: предельную квоту диска и порог предупреждения дисковой квоты. Например, можно задать для пользователя дисковую квоту в 500 мегабайт (МБ) и порог предупреждения дисковой квоты в 450 МБ. В этом случае пользователь сможет хранить на соответствующем томе не более 500 МБ файлов. Систему дисковых квот можно настроить таким образом, чтобы при сохранении пользователем на томе более 450 МБ файлов создавалась запись о

событии системы. Для управления квотами на томе необходимо входить в состав группы «Администраторы».

Можно разрешить пользователям превышать заданные квоты. Включение квот без ограничения использования дискового пространства полезно в случаях, когда не требуется запрещать пользователям доступ к тому, но требуется отслеживать использование дискового пространства отдельными пользователями. Также можно включить или отключить режим регистрации событий превышения пользователями заданных для них квот или порогов предупреждения. Отслеживание использования тома всеми пользователями начинается автоматически с момента включения дисковых квот для тома.

Квоты можно включать на локальных томах, сетевых томах и съемных дисках с файловой системой NTFS. Кроме того, для сетевых томов должен быть предоставлен общий доступ к корневому каталогу тома, а съемные диски должны быть предоставлены для общего доступа. Нельзя использовать сжатие файлов для предотвращения превышения пользователями заданных квот, поскольку сжатые файлы отслеживаются по их несжатому размеру.

При расчете использования тома сжатыми папками Windows, наоборот, использует размер папок после сжатия. Например, если папку размером 500 МБ сжать до 300 МБ, Windows сопоставит с квотой 300 МБ.

Включить дисковые квоты.

1. Щелкните правой кнопкой значок тома, для которого требуется включить дисковые квоты, и выберите команду Свойства.
2. В диалоговом окне Свойства откройте вкладку Квота.
3. Установите флажок Включить управление квотами на вкладке Квота.
4. Выберите один или несколько из следующих параметров и нажмите кнопку ОК:

- Не выделять место на диске при превышении квоты

Пользователи, превысившие квоту, получают сообщение об ошибке Windows "Недостаточно места на диске" и не могут записывать дополнительные данные в том без предварительного удаления или перемещения некоторых существующих файлов с диска.

В отдельных приложениях предусмотрен особый порядок действий в данной ситуации. Ситуация воспринимается приложением как переполнение диска. Если снять данный флажок, пользователи не смогут превышать предельную квоту. Включение квот без ограничения использования дискового пространства используется в случаях, когда не требуется запрещать пользователям доступ к тому, но требуется отслеживать использование дискового пространства отдельными пользователями. Можно также включить или отключить режим регистрации событий превышения пользователями заданных для них квот или порогов предупреждения.

- Выделять на диске не более

Укажите объем дискового пространства, выделяемого новым пользователям тома, а также порог, по достижении которого в системный журнал будет записано событие. Администраторы могут просматривать эти события в окне просмотра событий. Можно использовать десятичные числа (например, 20,5). Для дискового пространства и порога предупреждения в раскрывающемся списке выберите соответствующие наименования величин (например, Кбайт, Мбайт, Гбайт и т.п.).

- Регистрация превышения квоты пользователем

Если квоты включены, при превышении пользователями заданной предельной квоты в системный журнал на локальном компьютере заносится событие. Администраторы могут просматривать эти события в окне просмотра событий, отбирая их по типу.

По умолчанию события квоты записываются в системный журнал на локальном компьютере каждый час. Интервал записи событий квоты в системный журнал на локальном компьютере можно изменить с помощью команды fsutil behavior.

- Регистрация превышения порога предупреждения

Если квоты включены, при превышении пользователями заданного порога предупреждения в системный журнал на локальном компьютере заносится событие. Администраторы могут просматривать эти события в окне просмотра событий, отбирая их по типу.

По умолчанию события квоты записываются в системный журнал на локальном компьютере каждый час. Интервал записи событий квоты в системный журнал на локальном компьютере можно изменить с помощью команды fsutil behavior.

Контрольные вопросы:

1. Что такое аутентификация и идентификация?
2. Для чего применяются эти механизмы?
3. Что можно настроить с помощью вкладки Локальные политики безопасности?

Практическое занятие 6 Групповая политика. Политика аудита

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Теоретические сведения

Групповая политика Параметры групповой политики определяют различные компоненты окружения пользовательского рабочего стола, которыми управляет системный администратор (например, программы, доступные пользователям; программы, отображающиеся на пользовательском рабочем столе, и параметры меню **Пуск**). Чтобы создать конфигурацию рабочего стола для определенной группы пользователей используется оснастка «Групповая политика». Указанные параметры групповой политики содержатся в объекте групповой политики, который в свою очередь связан с выбранными объектами Active Directory — сайтами, доменами или подразделениями.

Конфигурация пользователя

Папка «Конфигурация пользователя» оснастки Групповая политика используется для задания политик, применяемых к пользователям независимо от того, какой компьютер используется для входа в систему.

Обычно узел «Конфигурация пользователя» содержит подпапки «Конфигурация программ», «Конфигурация Windows» и «Административные шаблоны», но поскольку оснастка «Групповая политика» имеет расширения, которые можно добавлять и удалять, то точный набор подпапок может различаться.

Конфигурация компьютера

С помощью узла «Конфигурация компьютера» в Групповой политике администраторы могут устанавливать политики, применяемые к компьютерам, вне зависимости от того, кто работает на них.

Узел «Конфигурация компьютера» обычно содержит подузлы «Конфигурация программ», «Конфигурация Windows» и «Административные шаблоны». Однако можно удалять и добавлять расширения групповой политики, поэтому подузлы могут отличаться от описанных выше.

Чтобы обновить групповую политику немедленно

1. Нажмите кнопку **Пуск** и выберите команду **Выполнить**.
2. В поле **Открыть** введите **gpupdate** и нажмите кнопку **ОК**.

2. Административные шаблоны:

В Windows включен ряд файлов .adm. Эти текстовые файлы, называемые административными шаблонами, содержат сведения о политике для элементов, расположенных в папке «Административные шаблоны» в дереве консоли оснастки Групповая политика.

Файлы .adm

Файл .adm состоит из иерархии категорий и подкатегорий, которые вместе

определяют отображение параметров политики. Кроме того, в файле содержатся следующие сведения:

- размещение параметров реестра, соответствующих каждому параметру;
- величина параметров или ограничений, связанных с каждым параметром;
- значение по умолчанию для большинства параметров;
- объяснение функции каждого параметра;
- версии Windows, поддерживающие каждый параметр.

Узел групповой политики «Административные шаблоны» содержит все сведения о политике на основе реестра. Конфигурация пользователя сохраняется в разделе **HKEY_CURRENT_USER** (HKCU), а конфигурация компьютера — в разделе **HKEY_LOCAL_MACHINE** (HKLM). В обоих этих разделах данные реестра, относящиеся к групповой политике, содержатся в папке \Software\Policies или \Software\Microsoft\Windows\CurrentVersion\Policies. Следовательно, параметры групповой политики хранятся в реестре в четырех областях.

Чтобы добавить или удалить файл административного шаблона (.adm):

1. Откройте редактируемый объект групповой политики и в дереве консоли щелкните правой кнопкой папку **Административные шаблоны**.
2. Выберите команду **Добавление и удаление шаблонов**.
3. Для удаления шаблона в списке **Текущие шаблоны политики** выберите шаблон и нажмите кнопку **Удалить**.

Если необходимо добавить шаблон, нажмите кнопку **Добавить**. В диалоговом окне **Шаблоны политики** щелкните шаблоны, которую требуется добавить, и нажмите кнопку **Открыть**.

4. В диалоговом окне **Добавление и удаление шаблонов** нажмите кнопку **Закрыть**.
3. **Политика аудита:**

Перед внедрением аудита необходимо выбрать политику аудита. Политика аудита указывает категории событий для аудита, связанных с безопасностью. При первой установке Windows XP Professional все категории аудита выключены. Включая аудит различных категорий событий, можно создавать политику аудита, удовлетворяющую всем требованиям организации.

Для проведения аудита можно выбрать следующие категории событий.

- Аудит событий входа в систему
- Аудит управления учетными записями
- Аудит доступа к службе каталогов
- Аудит входа в систему
- Аудит доступа к объектам
- Аудит изменения политики
- Аудит использования привилегий
- Аудит отслеживания процессов
- Аудит системных событий

Задание.

1. Создать оснастку Групповая политика и сохранить ее на рабочий стол.
2. Запретить пользователям шифровать данные, используя EFS
3. Установить политику аудита на успех входа/выхода из системы; изменение политики
4. Запретить администраторам изменять системное время
5. Скрыть команду "Свойства" в контекстном меню объекта "Мой компьютер"
6. Установить блокировку учетной записи пользователя на 5 минут при троекратном неверном вводе пароля.
7. Установить доступ к компьютеру из сети только администраторам

Контрольные вопросы

1. Что представляют собой групповые политики?
2. Дя чего используются групповые политики?
3. На какие области разделена утилита Group Policies?
4. Групповые политики по умолчанию
5. Дополнения групповой политики в Windows
6. Средства управления групповой политикой
7. Управление пользователями и группами AD
8. Оснастка Active Directory Users and Computers (Пользователи и компьютеры Active Directory)

9. Какова роль аудита в обеспечении безопасности компьютерной системы?
10. Где и каким образом формируется информация о событиях аудита?
11. Какая информация может быть получена в результате аудита?
12. Какие типы аудита вы знаете и для чего предназначен каждый из них?
13. Каким образом активизируется политика аудита?
14. Каким образом политика аудита применяется для выбранных объектов и пользователей?
15. В каких случаях целесообразно учитывать *Успех*, а когда целесообразно фиксировать *Отказ?*
16. Как пользоваться журналами безопасности?
17. Какие учетные записи дают право на настройку аудита и проверку результатов аудита? Каким образом администратор может использовать информацию об аудите для повышения безопасности системы?

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6.1.1. Основная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л1.1	Башлы П. Н., Бабаш А. В., Баранова Е. К.	Информационная безопасность и защита информации: Учебное пособие	Москва: Евразийский открытый институт, 2012	http://www.iprbookshop.ru/10677.html
6.1.2. Дополнительная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л2.1	В.В. Горгорова, А.В. Чернов	Информационная безопасность: учебное пособие	, 2011	https://ntb.donstu.ru/content/informacionnaya-bezopasnost
	Авторы,	Заглавие	Издательство, год	Адрес
Л2.2	Прохорова О. В.	Информационная безопасность и защита информации: Учебник	Самара: Самарский государственный архитектурно-строительный университет, ЭБС АСВ, 2014	http://www.iprbookshop.ru/43183.html
6.1.3. Методические разработки				
	Авторы,	Заглавие	Издательство, год	Адрес

Л3.1	ДГТУ, Каф. "ВСИиБ"; сост. В.В. Галушка	Методические указания к лабораторным работам по дисциплине «Информационная безопасность телекоммуникационных систем»	Ростов н/Д.: ИЦ ДГТУ, 2018	https://ntb.donstu.ru/content/metodicheskie-ukazaniya-k-laboratornym-rabotam-po-discipline-informacionnaya-bezopasnost-telekommunikacionnyh-sistem

6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"	
Э1	Артемов А.В. Информационная безопасность [Электронный ресурс]: курс лекций/ Артемов А.В.— Электрон. текстовые данные.— Орел: Межрегиональная Академия безопасности и выживания (МАБИВ), 2014.— 256 c.http://www.iprbookshop.ru/33430.html
Э2	Башлы П.Н. Информационная безопасность и защита информации [Электронный ресурс]: учебное пособие/ Башлы П.Н., Бабаш А.В., Баранова Е.К.— Электрон. текстовые данные.— М.: Евразийский открытый институт, 2012.— 311 c.http://www.iprbookshop.ru/10677
Э3	Галатенко В.А. Основы информационной безопасности [Электронный ресурс]/ Галатенко В.А.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 266 c.http://www.iprbookshop.ru/52209
6.3.1 Перечень программного обеспечения	
6.3.1.1	ОС Windows ;
6.3.1.2	Kaspersky Endpoint Security ;
6.3.1.3	Microsoft Office 2007 Professional Plus
6.3.1.4	Borland Developer Studio 2006
6.3.2 Перечень информационных справочных систем	
6.3.2.1	1. ЭБС «Консультант студента. Электронная библиотека» http://www.studmedlib.ru/ru
6.3.2.2	2. Профессиональные справочные системы "Техэксперт" http://www.cntd.ru/

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Модели и методы планирования экспериментов»
для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Модели и методы планирования экспериментов» содержат задания для студентов, необходимые для лабораторных занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины - развить системное мышление у обучающихся путем детального анализа подходов к математическому моделированию и сравнительного анализа разных типов моделей. Ознакомить обучающихся с математическими свойствами методов и моделей оптимизации, которые могут использоваться при анализе и решении широкого спектра задач. Выработать у обучающихся навыки проведения численных исследований математических моделей и анализа результатов вычислений. Научить выбирать наиболее перспективное управляющее решение.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

ОПК-8.3: Владеет приемами разработки программных средств и проектов, командной работы.

ОПК-7.1: Использует математические алгоритмы функционирования, принципы построения, модели хранения и обработки данных распределенных информационных систем и систем поддержки принятия решений

ОПК-2.2: Обосновывает выбор современных информационно-коммуникационных и интеллектуальных технологий, разрабатывать оригинальные программные средства для решения профессиональных задач

ОПК-2.3: Разрабатывает оригинальные программные средства, в том числе оригинальных программных средств, в том числе с использованием современных информационно-коммуникационных и интеллектуальных технологий, для решения профессиональных задач

Изучив данный курс, студент должен:

Знать:

основные понятия и принципы планирования эксперимента, критерии оптимальности, разновидности и правила построения планов экспериментов;

основы корреляционного, дисперсионного и регрессионного анализа;

методы оптимизации многофакторных объектов.

Уметь:

проводить статистическую оценку результатов экспериментов и применять различные критерии согласия для проверки статистических гипотез;

выбирать план эксперимента, исходя из имеющихся возможностей и целей эксперимента;

проводить оптимизацию объекта исследования.

Владеть:

планирования на основе теории эксперимента при решении различных инженерных задач;

выполнения корреляционного, регрессионного и дисперсионного анализов с привлечением стандартных программных пакетов.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному

специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическое занятие № 1
Составление плана полного факторного эксперимента

Цель занятия:

-приобретение навыков в составлении плана полного факторного эксперимента.

Учебные вопросы.

1 Термины и определения.

2 Составление плана.

Основные теоретические сведения

Полный факторный эксперимент

На начальных этапах оптимизации для определения градиента применяют неполные полиномы второго порядка или линейные полиномы. Вычисление оценок коэффициентов таких полиномов осуществляется на основе обработки результатов реализации наиболее простых планов, в которых каждый фактор принимает только два значения $v_{i,\,min}$ или $v_{i,\,max}$, расположенные симметрично относительно некоторого нулевого уровня или центра плана по данному фактору. Значения уровней варьирования выбирает исследователь, исходя из возможного диапазона изменения каждого фактора и возможности применения линейной аппроксимации функции отклика в выбранном диапазоне изменений параметра. Без ограничения общности можно считать, что кодированные значения x_i принимают значения – 1 и +1 соответственно (или просто – или +). Множество всех точек в k-мерном пространстве, координаты которых являются комбинациями "+" и "–", называется полным факторным планом или планом *полного факторного эксперимента* типа 2^k (ПФЭ). Количество точек в этом плане $N = 2^k$.

Для примера возьмем полный факторный эксперимент с тремя независимыми переменными x_1, x_2 и x_3, показанный в таблице 1.

Таблица 1

Матрица планирования								Вектор результатов
x_0	x_1	x_2	x_3	$x_1 x_2$	$x_1 x_3$	$x_2 x_3$	$x_1 x_2 x_3$	y
+	–	–	–	+	+	+	–	y_1
+	–	–	+	+	–	–	+	y_2
+	–	+	–	–	+	–	+	y_3
+	–	+	+	–	–	+	–	y_4
+	+	–	–	–	–	+	+	y_5
+	+	–	+	–	+	–	–	y_6
+	+	+	–	+	–	–	–	y_7
+	+	+	+	+	+	+	+	y_8

Второй, третий и четвертый столбцы таблицы соответствуют собственно плану экспериментов, пятый – восьмой столбцы содержит значения произведений независимых переменных. Фиктивная переменная $x_0 = 1$ (первый столбец) введена для единообразия записи расчетных формул коэффициентов полинома. Строки соответствуют опытам, например, первая строка характеризует эксперимент, в котором все независимые переменные находятся на нижнем уровне.

Существует несколько способов построения подобных матриц планирования. В частности можно воспользоваться приемом, характерным для записи последовательности двоичных чисел. В столбце последней переменной x_3 знаки меняются поочередно, в столбце предпоследней переменной x_2 – чередуются через два элемента, третьей справа переменной x_1 – через четыре элемента. Аналогично строится матрица для любого количества переменных, порядок перечисления переменных не играет роли. Столбцы с

произведениями переменных вычисляются путем умножения значений элементов в соответствующих столбцах простых переменных.

Из анализа матрицы планирования легко видеть, что полный факторный эксперимент обладает свойствами:

ортогональности. Сумма парных произведений элементов любых двух различных столбцов равна нулю. В частности, для простых переменных

$$\sum_{u=1}^{N} x_{iu} x_{ju} = 0, \quad i \neq j, \; i, j = \overline{0,k};$$

симметричности. Сумма всех элементов любого столбца, за исключением первого, равна нулю, например

$$\sum_{u=1}^{N} x_{iu} = 0, \quad i = \overline{1,k};$$

нормированности. Сумма квадратов элементов любого столбца равна числу опытов, так для *i*-й переменной

$$\sum_{u=1}^{N} x_{iu}^2 = N, \quad i = \overline{0,k}$$

Первые два свойства обеспечивают независимость оценок коэффициентов модели и допустимость их физической интерпретации. Нарушение этих свойств приводит к взаимной зависимости оценок и невозможности придания смысла коэффициентам.

Включение в матрицу планирования переменных вида x_i^2 приведет к появлению единичных столбцов, совпадающих друг с другом и со столбцом x_0. Следовательно, нельзя будет определить, за счет чего получено значение □$_0$. Поэтому планы ПФЭ 2^k не применимы для построения функции отклика в виде полного полинома второй степени.

Методические рекомендации по выполнению практического занятия

Изучить теоретический материал и построить полный факторный эксперимент для исследования модели предметной области магистерской диссертации.

Контрольные вопросы

Практическое занятие № 2
Составление плана дробного факторного эксперимента

Цель занятия:

-приобретение навыков в составлении плана дробного факторного эксперимента.

Учебные вопросы.

1 Термины и определения.

2 Составление плана.

Основные теоретические сведения

Дробный факторный эксперимент

С ростом количества факторов k число точек плана в ПФЭ растет по показательной функции 2^k. Планы ПФЭ позволяют получить несмещенные оценки градиента функции отклика в центральной точке, но в случае применения линейного полинома оказываются недостаточно эффективными по количеству опытов при большом числе независимых переменных, так как остается слишком много степеней свободы на проверку адекватности модели. Например, при $k = 5$ на проверку адекватности линейной модели остается 26 степеней. Хотя большое количество опытов и приводит к существенному снижению

погрешности в оценке коэффициентов, все же такое число степеней свободы для проверки адекватности является чрезмерным.

Таким образом, в случаях, когда используются только линейные приближения функции отклика, количество опытов следует сократить, используя для планирования так называемые *регулярные дробные реплики* от ПФЭ, содержащие подходящее число опытов и сохраняющие основные свойства матрицы планирования. Реплика, включающая только половину экспериментов ПФЭ, называется полурепликой, включающая четвертую часть опытов – четвертьрепликой и т. д. Краткое обозначение указанных дробных реплик 2^{k-1}, 2^{k-2} соответственно.

Построение регулярной дробной реплики или проведение *дробного факторного эксперимента* (ДФЭ) типа 2^{k-p} предусматривает отбор из множества k факторов $k-p$ основных, для которых строится план ПФЭ. Этот план дополняется p столбцами, которые соответствуют остальным факторам. Каждый из этих столбцов формируется по специальному правилу, а именно, получается как результат поэлементного умножения не менее двух и не более $k-p$ определенных столбцов, соответствующих основным факторам. Иначе говоря, в дробных репликах p линейных эффектов приравнены к эффектам взаимодействия. Но именно такое построение матрицы планирования и позволяет обеспечить ее симметричность, ортогональность и нормированность.

Таблица 3.2

Матрица планирования				Вектор результатов
x_0	x_1	x_2	x_3	y
+	–	–	+	y_1
+	–	+	–	y_2
+	+	–	–	y_3
+	+	+	+	y_4

Правило образования каждого из p столбцов ДФП называют *генератором плана*. Каждому дополнительному столбцу соответствует свой генератор (для плана типа 2^{k-p} должно быть задано p различных генераторов). Генератор задается как произведение основных факторов, определяющее значение элементов соответствующего дополнительного столбца матрицы планирования. Примером записи генератора для плана 2^{3-1} служит выражение $x_3 = x_1x_2$, табл. 3.2. Матрица планирования ДФП типа 2^{k-p} содержит $k + 1$ столбец и $N = 2^{k-p}$ строк.

Методические рекомендации по выполнению практического занятия

Изучить теоретический материал и построить дробный факторный эксперимент для исследования модели предметной области магистерской диссертации.

Контрольные вопросы

Практическое занятие № 1
Оценка коэффициентов функции отклика

Цель занятия:

-приобретение навыков в составлении плана эксперимента.

Учебные вопросы.

1 Термины и определения.
2 Технические процессы.
3 Стадии жизненного цикла.

Основные теоретические сведения

Оценки коэффициентов функции отклика

С помощью матрицы планирования, описанной в табл. 3.1, можно вычислить оценки коэффициентов неполного полинома третьей степени

$$y' = \square_0 + \square_1 x_1 + \square_2 x_2 + \square x_3 + \square_{12} x_1 x_2 + \square_{13} x_1 x_3 + \square_{23} x_2 x_3 + \square_{123} x_1 x_2 x_3$$

или линейной функции

$y = \square_0 + \square_1 x_1 + \square_2 x_2 + \square_3 x_3$.

Первый вид полинома позволяет оценить не только влияние отдельных факторов, но и один из часто встречающихся видов нелинейности, когда эффект одного фактора зависит от уровня других факторов, т.е. присутствует эффект взаимодействия факторов. Эффект взаимодействия вида $x_i\, x_j$ называют парным, $x_i\, x_j\, x_k$ – тройным и т. д. С ростом количества факторов число возможных взаимодействий быстро увеличивается. Суммарно количество всех коэффициентов функции отклика такого типа равно числу опытов полного факторного эксперимента.

Оценки коэффициентов полинома определяются на основе метода наименьших квадратов и для рассматриваемого типа ПФЭ вычисляются по простым соотношениям [8, стр. 29]

$$\beta_i = \frac{1}{N}\sum_{u=1}^{N} x_{iu}\bar{y}_u, \quad i = \overline{0,k}; \qquad \beta_{i,\ldots,m} = \frac{1}{N}\sum_{u=1}^{N} x_{iu}\ldots x_{mu}\bar{y}_u, \quad i = \overline{1,k}, \quad m > i. \tag{3.1}$$

Здесь величина y соответствует значению отклика в указанной точке факторного пространства при отсутствии повторных опытов или является оценкой математического ожидания

$$\bar{y}_u = \frac{1}{r_u}\sum_{i=1}^{r_u} y_{ui}$$

значений функции отклика по всем r_u повторным опытам в данной точке. Повторные опыты проводятся в тех случаях, когда на функционирование системы оказывают влияние случайные воздействия. Количество повторных опытов в разных точках плана может различаться.

Допустима следующая интерпретация оценок коэффициентов:

$\square_0$ соответствует значению функции отклика в центре проводимого эксперимента;

$\square_i$ равен приращению функции при переходе значения фактора i с нулевого уровня на верхний (это вклад соответствующего фактора в значение функции);

$\square_{ij}$ равен нелинейной части приращения функции при одновременном переходе факторов i и j с нулевого уровня на верхний и т.п.

Ошибки в определении коэффициентов полинома можно охарактеризовать соответствующей дисперсией. С учетом того, что кодированные значения факторов принимают значения +1 и – 1, оценка дисперсии коэффициента определяется соотношением

$$\sigma^2(\beta_i) = D\left[\frac{1}{N}\sum_{u=1}^{N} x_u \bar{y}_u\right] = \frac{1}{N^2}\sum_{u=1}^{N} D(\bar{y}_u) = \frac{1}{N}\left[\frac{1}{N}\sum_{u=1}^{N} D(\bar{y}_u)\right]. \tag{3.2}$$

Следовательно, оценка дисперсии всех коэффициентов одинакова и определяется только дисперсией средних значений функции отклика и числом опытов. Эту формулу можно применять, если количество опытов во всех точках плана одинаково. При факторном эксперименте, в отличие от классического, одновременно варьируются все

факторы, поэтому каждый коэффициент полинома определяется по результатам всех экспериментов, тем самым оценка дисперсии коэффициентов получается в N раз меньше средней дисперсии всех опытов. Оценка дисперсии среднего значения в конкретной точке плана

$$D(\bar{y}_u) = \sigma_u^2 / r_u ,$$

где σ_u^2 – оценка дисперсии функции отклика в точке u, r_u – число повторных опытов в этой точке плана [7, стр. 50]. Дисперсия оценок всех коэффициентов одинакова, поэтому ПФЭ рассмотренного типа являются ротатабельным.

При использовании неполных полиномов k-го порядка количество точек плана равно количеству оцениваемых параметров (насыщенное планирование). Поэтому не остается степеней свободы для проверки гипотезы об адекватности представления результатов эксперимента заданной математической моделью. Если применять полиномы первой степени, то тогда остаются степени свободы для проверки гипотезы об адекватности модели.

Методические рекомендации по выполнению практического занятия

Изучить теоретический материал и выполнить оценку коэффициентов функции отклика дробного факторного эксперимента для исследования модели предметной области магистерской диссертации.

Контрольные вопросы

Практическое занятие № 4
Предварительная обработка результатов эксперимента

Цель занятия:

-приобретение навыков в предварительной обработке результатов эксперимента.

Учебные вопросы.

1 Термины и определения.
2 Выполнение предварительной обработки

Основные теоретические сведения

Предварительная обработка результатов эксперимента

После того, как составлен план проведения эксперимента, можно приступать к его проведению. Вопросы непосредственного осуществления эксперимента рассматривать не будем, а перейдем к обработке результатов. Сущность обработки результатов эксперимента во многом одинакова для различных областей применения – поиска оптимума функции, описания поверхности отклика и др.

Необходимо учитывать, что любой эксперимент сопровождается погрешностями (методическими, измерений) и содержит элементы неопределенности (случайности). Проведение повторных опытов не дает полностью совпадающих результатов. Поэтому процедура обработки должна учитывать эти обстоятельства. Обработка результатов включает предварительную обработку результатов экспериментов, вычисление оценок коэффициентов функции отклика и проведение ряда проверок: однородности дисперсии воспроизводимости, адекватности модели и значимости коэффициентов [2, 5, 6]. Расчетные соотношения будут приведены в предположении, что в каждой точке плана производиться различное количество повторных опытов r_u.

В ходе предварительной обработки вычисляются следующие параметры для всех точек $u = \overline{1, N}$ плана экспериментов:

среднее значение функции отклика

$$\bar{y}_u = \frac{1}{r_u}\sum_{i=1}^{r_u} y_{u\,i}$$;

несмещенная оценка дисперсии функции отклика

$$\sigma_u^2 = \frac{1}{r_u - 1}\sum_{i=1}^{r_u}\left(y_{ui} - \bar{y}_u\right)^2$$.

Для данной величины количество степеней свободы $\nu_u = r_u - 1$;

оценка дисперсии среднего значения функции отклика (*оценка дисперсия воспроизводимости*)

$$D(\bar{y}_u) = \sigma_u^2 / r_u = D_u$$.

На основе частных оценок вычисляется средняя величина оценки *дисперсии воспроизводимости среднего значения функции отклика* по всей области планирования

$$\sigma^2(y) = \left\{\sum_{u=1}^{N}(r_u - 1)D_u\right\} / \left\{\sum_{u=1}^{N}(r_u - 1)\right\} = \left\{\sum_{u=1}^{N}(r_u - 1)D_u\right\} / \left\{\sum_{u=1}^{N} r_u - N\right\} \qquad (4.1)$$.

Эта оценка является несмещенной и ее можно рассматривать как случайную величину с количеством степеней свободы

$$\varphi(y) = \sum_{u=1}^{N} r_u - N$$.

Именно величину $\sigma^2(y)$ следует использовать как оценку дисперсии воспроизводимости среднего значения функции отклика вместо

$$\frac{1}{N}\sum_{u=1}^{N} D(\bar{y}_u)$$

в выражении (3.2).

Методические рекомендации по выполнению практического занятия

Изучить теоретический материал и выполнить оценку коэффициентов функции отклика дробного факторного эксперимента для исследования модели предметной области магистерской диссертации

Контрольные вопросы

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6.1.1. Основная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л1.1	Порсев, Е. Г.	Организация и планирование экспериментов: учебное пособие	Новосибирск: Новосибирский государственный технический университет, 2010	http://www.iprbookshop.ru/45415.html
Л1.2	Ермаков, А. С.	Планирование и организация эксперимента: методические указания к практическим занятиям для студентов, обучающихся по направлению подготовки 221700 «стандартизация и метрология»	Москва: Московский государственный строительный университет, Ай Пи Эр Медиа, ЭБС АСВ, 2014	http://www.iprbookshop.ru/25512.html
6.1.2. Дополнительная литература				
	Авторы,	Заглавие	Издательство, год	Адрес
Л2.1	Сагдеев, Д. И.	Основы научных исследований, организация и планирование эксперимента: учебное пособие	Казань: Казанский национальный исследовательский технологический университет, 2016	http://www.iprbookshop.ru/79455.html
Л2.2	Емельянов, А. М., Кидяева, Н. П., Подолько, Е. А., Шпилев, Е. М.	Статистические методы обработки, планирования инженерного эксперимента: учебное пособие	Благовещенск: Дальневосточный государственный аграрный университет, 2015	http://www.iprbookshop.ru/55912.html
6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"				
Э1	Сафин, Р. Г. Основы научных исследований. Организация и планирование эксперимента [Электронный ресурс] : учебное пособие / Р. Г. Сафин, А. И. Иванов, Н. Ф. Тимербаев. — Электрон. текстовые данные. — Казань : Казанский национальный исследовательский технологический университет, 2013. — 154 с. — 978-5-7882-1412-2. — Режим доступа: http://www.iprbookshop.ru/62219.html			
Э2	Бойко, А. Ф. Теория планирования многофакторных экспериментов [Электронный ресурс] : учебное пособие / А. Ф. Бойко, М. Н. Воронкова. — Электрон. текстовые данные. — Белгород : Белгородский государственный технологический университет им. В.Г. Шухова, ЭБС АСВ, 2013. — 73 с. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/28403.html			
Э3	Горохов, В. Л. Планирование и обработка экспериментов [Электронный ресурс] : учебное пособие / В. Л. Горохов, В. В. Цаплин. — Электрон. текстовые данные. — СПб. : Санкт-Петербургский государственный архитектурно- строительный университет, ЭБС АСВ, 2016. — 88 с. — 978-5-9227-0608-7. — Режим доступа: http://www.iprbookshop.ru/63623.html			

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Теоретические основы программирования»
для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Теоретические основы программирования» содержат задания для студентов, необходимые для лабораторных занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины - развить системное мышление у обучающихся путем детального анализа подходов к математическому моделированию и сравнительного анализа разных типов моделей. Ознакомить обучающихся с математическими свойствами методов и моделей оптимизации, которые могут использоваться при анализе и решении широкого спектра задач. Выработать у обучающихся навыки проведения численных исследований математических моделей и анализа результатов вычислений. Научить выбирать наиболее перспективное управляющее решение.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

ОПК-8.3: Владеет приемами разработки программных средств и проектов, командной работы

ОПК-7.1: Использует математические алгоритмы функционирования, принципы построения, модели хранения и обработки данных распределенных информационных систем и систем поддержки принятия решений

ОПК-2.2: Обосновывает выбор современных информационно-коммуникационных и интеллектуальных технологий, разрабатывать оригинальные программные средства для решения профессиональных задач

ОПК-2.3: Разрабатывает оригинальные программные средства, в том числе оригинальных программных средств, в том числе с использованием современных информационно-коммуникационных и интеллектуальных технологий, для решения профессиональных задач

Изучив данный курс, студент должен:

Знать:

-языки спецификации программ;
-методы описания алгоритмов;
-методы доказательства правильности и оценки качества программ;
-методы доказательности правильности программ;
-методы верификации и валидации программ;
-критерии эффективности программ.

Уметь:

-разрабатывать стратегию и определять цели проектирования программ;
-применять языки спецификации программ;
-выполнять доказательства правильности программ;
-проводить исследование качества программ.
-проводить разработку моделей и алгоритмов решения задач;
-выполнять доказательства правильности программ;
-выполнять верификацию и валидацию программ;
-оценивать качество программ и их эффективность.

Владеть:

-экспериментально-теоретическими методами исследования;
-методами моделирование систем и процессов.
-методами разработки алгоритмов;
-методами верификации и валидации программ.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическое занятие 1

Анализ методов программирования

Цель работы:

1. Закрепить знания об основных методах программирования.
2. Научиться выбирать метод программирования.

1 Задание на лабораторную работу

1.1 Изучить прикладные методы программирования
1.2 Ознакомиться с теоретическими методами программирования

2 Краткие теоретические сведения

Ниже изложены базовые основы методов прикладного, систематического (структурного, компонентного, аспектно-ориентированного и др.) и теоретического (алгебраического, композиционного, концепторного и алгебро-алгоритмического) программирования для ознакомления студентов с теоретическими и прикладными аспектами методов программирования.

В последние годы наибольшее развитие и использование получили прикладные методы: объектно-ориентированный, компонентный, сервисно-ориентированный и др. Они составляют основу методологии разработки ПС и практически применяются при создании разных видов программ.

Находят применение формальные и теоретические методы программирования (алгебраический, алгебро-алгоритмический, композиционный и др.), которые основываются на математических, логико-алгоритмических и математических подходах.

В данной лекции представлено описание базовых понятий и особенностей методов систематического (прикладного), отдельных методов теоретического программирования с целью ознакомления студентов с современной теорией и практикой разработки программ.

2.1 Методы систематического программирования

К методам систематического программирования отнесены следующие методы:

- структурный,
- объектно-ориентированный,
- UML-метод,
- компонентный,
- аспектно-ориентированный,
- генерирующий,
- агентный и др.

Каждый метод имеет свое множество понятий и операций для проведения процесса разработки компонентов или ПС. Метод генерирующего программирования использует возможности объектно-ориентированного, компонентного, аспектно-ориентированного методов и др.

2.1.1 СТРУКТУРНЫЙ МЕТОД

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

В основе структурного метода лежат такие общие принципы:

- разбивка системы на множество независимых задач, доступных для понимания и решения;
- иерархическое упорядочивание, т.е. организация составных частей проблемы в древовидные структуры с добавлением новых деталей на каждом уровне.

К основным принципам относятся:

- абстрагирование, т.е. выделение существенных аспектов системы и отвлечение от несущественных;
- формализация, т.е. общее методологическое решение проблемы;
- непротиворечивость, состоящая в обосновании и согласовании элементов системы;
- иерархическая структуризация данных.

При структурном анализе применяются три наиболее распространенные модели проектирования ПС:

- SADT (Structured Analysis and Design Technique) - модель и соответствующие функциональные диаграммы;
- SSADM (Structured Systems Analysis and Design Method) - метод структурного анализа и проектирования;
- IDEF0 (Integrated Definition Functions) - метод создания функциональной модели, IDEF1 - информационной модели, IDEF2 - динамической модели и др..

На стадии проектирования эти модели расширяются, уточняются и дополняются диаграммами, отражающими структуру или архитектуру системы, структурные схемы программ и диаграммы экранных форм.

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

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

Основные элементы этого метода базируются на следующих концепциях:

- графическое представление структуры системы диаграммами, отображающими функции в виде блоков, а интерфейсы (вход/выход) - дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описывается в виде ограничений, которые определяют условия управления и выполнения функции;
- наличие ограничений на количество блоков (от 3 до 6) на каждом уровне декомпозиции и связей диаграмм через номера этих блоков;
- уникальность меток и наименований;
- разделение входов и операций управления;
- отделение управления от функций, т.е. исключение влияния организационной структуры на функциональную модель.

Метод SADT применяется при моделировании широкого круга систем, для которых определяются требования и функции, а затем проводится их реализация. Средства SADT могут применяться при анализе функций в действующей ПС, а также при определении способов их реализации.

Результат применения метода SADT - модель, которая состоит из диаграмм, фрагментов текстов и глоссария со ссылками друг на друга. Все функции и интерфейсы представляются диаграммами в виде блоков и дуг. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке, указывается с левой стороны блока, а результаты выхода - с правой стороны. Механизм, осуществляющий операцию (человек или автоматизированная система), задается дугой, входящей в блок снизу (рис. 5.1).

Одна из наиболее важных особенностей метода SADT - постепенная детализация модели системы по мере добавления диаграмм, уточняющих эту модель.

Метод SSADM базируется на таких структурных диаграммах, как последовательность, выбор и итерация. Моделируемый объект задается последовательностью групп, операторами выбора из группы и циклическим выполнением отдельных элементов.

Рисунок 1 - Структура модели

Базовая диаграмма - иерархическая и включает в себя: список компонентов описываемого объекта; идентифицированные группы выбранных и повторяемых компонентов, а также последовательно используемых компонентов.

Данный метод представлен моделью ЖЦ со следующими этапами разработки программного проекта (рис. 2):

- стратегическое проектирование и изучение возможности выполнения проекта;
- детальное обследование предметной области, включающее в себя анализ и спецификацию требований;
- логическое проектирование и спецификация системы;
- физическое проектирование структур данных в соответствии с выбранной структурой БД (иерархической, сетевой и др.);
- конструирование и тестирование системы.

Детальное обследование предметной области проводится для того, чтобы изучить ее особенности, рассмотреть потребности и предложения заказчика, провести анализ требований из разных документов, специфировать их и согласовать с заказчиком.

Цель стратегического проектирования - определение области действия проекта, анализ информационных потоков, формирование общего представления об архитектуре системы, затратах на разработку и подтверждение возможности дальнейшей реализации проекта. Результат - спецификация требований, которая применяется при разработке логической структуры системы.

Логическое проектирование - это определение функций, диалога, метода построения и обновления БД. В логической модели отображаются входные и выходных данные, прохождение запросов и установка связей между сущностями и событиями.

Рисунок 2 - Жизненный цикл SSADM

Физическое проектирование - это определение типа СУБД и представления данных в ней с учетом спецификации логической модели данных, ограничений на память и времени обработки, а также определение механизмов доступа, размера логической БД, связей между элементами системы. Результат - создание документа, включающего в себя:

- спецификацию функций и способов их реализации, описание процедурных, непроцедурных компонентов и интерфейсов системы;
- определение логических и физических групп данных с учетом структуры БД, ограничений на оборудование и положений стандартов на разработку;
- определение событий, которые обрабатываются как единое целое и выдача сообщений о завершении обработки и др.

Конструирование - это программирование элементов системы и их тестирование на наборах данных, которые подбираются на ранних этапах ЖЦ разработки системы.

Проектирование системы является управляемым и контролируемым. Создается сетевой график, учитывающий работы по разработке системы, затраты и сроки. Слежение и контроль выполнения плана проводит организационный отдел. Проект системы задается структурной моделью, в которой содержатся работы и взаимосвязи между ними и их исполнителями, а потоки проектных документов между этапами отображаются в сетевом графике. Результаты каждого из этапов ЖЦ контролируются и передаются на следующий этап в виде, удобном для дальнейшей реализации другими исполнителями.

2.1.2 ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ МЕТОД

Объектно-ориентированный подход (ООП) - стратегия разработки, в рамках которой разработчики системы вместо операций и функций мыслят *объектами* . Объект - это предмет внешнего мира, некоторая сущность, пребывающая в различных состояниях и имеющая множество операций. Операции задают сервисы, предоставляемые объектам для выполнения определенных вычислений, а состояние - это набор атрибутов объекта, поведение которых изменяет это состояние.

Объекты группируются в класс, который служит шаблоном для включения описания всех атрибутов и операций, связанных с объектами данного класса.

Программная система содержит взаимодействующие объекты, имеющие собственное локальное состояние и набор операций для определения состояний других

объектов. Объекты скрывают информацию о представлении состояний и ограничивают к ним доступ.

Под *процессом* в ООП понимается проектирование классов объектов и взаимоотношений между ними (рис. 3).

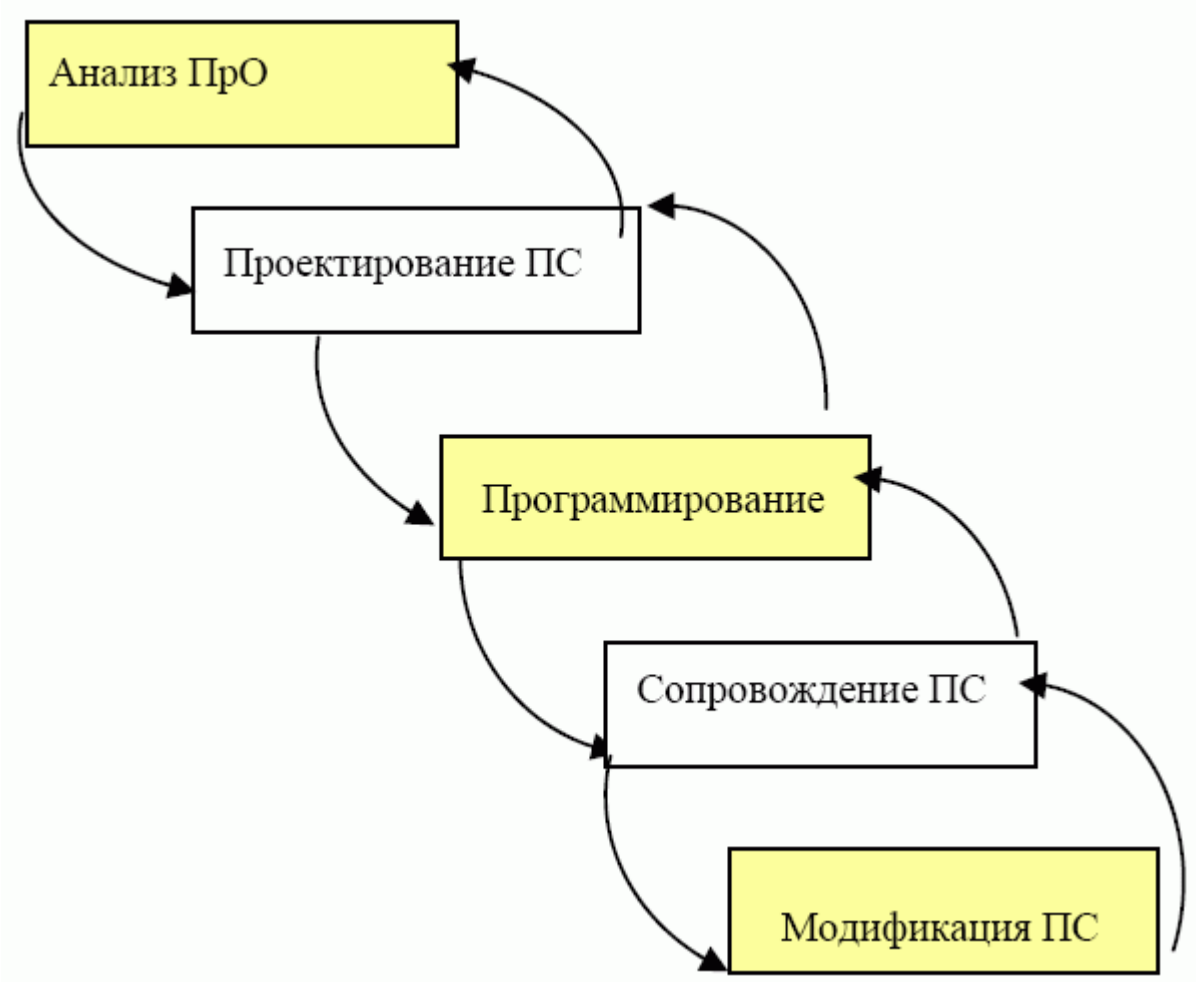


Рисунок 3 - ЖЦ разработки ПС в среде ООП

Процесс разработки включает в себя следующие этапы:

- *анализ* – создание объектной модели (ОМ) ПрО, в которой объекты отражают реальные ее сущности и операции над ними;
- *проектирование* – уточнение ОМ с учетом описания требований для реализации – реализация ОМ средствами языков программирования C++, Java и др.
- *сопровождение* – конкретных задач системы;
- *программирование* использование и развитие системы, внесение изменений, как в состав объектов, так и в методы их реализации;
- *модификация ПС* – изменение системы в процессе ее сопро-вождения путем добавления новых функциональных возможностей, интерфейсов и операций.

Приведенные этапы могут выполняться итерационно друг за другом и с возвратом к предыдущему этапу. На каждом этапе может применяться одна и та же система нотаций.

Переход к следующему этапу приводит к усовершенствованию результатов предыдущего этапа путем более детальной реализации ранее определенных классов объектов и добавления новых классов.

Результат процесса анализа ЖЦ - модель ПрО и набор других моделей (модель архитектуры, модель окружения и использования), полученных на этапах процесса ЖЦ. Модели отображают связи между объектами, их состояния и набор операций для динамического изменения состояния других объектов, а также взаимоотношения со средой. Объекты инкапсулируют информацию об их состоянии и ограничивают к ним

доступ. Модели окружения и использования системы - это две взаимно дополняющие друг друга модели связи системы со средой.

Модель окружения системы - статическая модель, которая описывает другие подсистемы из пространства разрабатываемой ПС, а модель использования системы - динамическая модель, которая определяет взаимодействие системы со своей средой. Это взаимодействие определяется последовательностью запросов к сервисам объектов и ответных реакцией системы после выполнения запроса. После определения взаимодействий между объектами проектируемой системы и ее окружением, полученные данные используются для разработки архитектуры системы из объектов, созданных в предыдущих подсистемах и проектах.

Существует два типа моделей системной архитектуры:

- *статическая модель* описывает статическую структуру системы в терминах классов объектов и взаимоотношений между ними (обобщение, расширение, использование, структурные отношения);
- *динамическая модель* описывает динамическую структуру системы и взаимодействие между объектами во время выполнения системы.

Результат проектирования - это ПС, в которой определены все необходимые объекты статически или динамически с помощью классов и соответствующих методов реализации объектов. Полученная объектно-ориентированная система проверяется на показатели качества на основе результатов тестирования и сбора данных об ошибках и отказах системы. Такую систему можно рассматривать как совокупность автономных и независимых объектов. Изменение метода реализации объекта или добавление новых функций не влияет на другие объекты системы. Объекты могут быть повторно используемыми.

2.1.3 UML-МЕТОД МОДЕЛИРОВАНИЯ

UML (United Modeling Language) - унифицированный язык моделирования является результатом совместной разработки специалистов программной инженерии и инженерии требований. Он широко используется ведущими разработчиками ПО как метод моделирования на этапах ЖЦ разработки ПС.

В основу метода положена парадигма объектного подхода, при котором концептуальное моделирование проблемы происходит в терминах взаимодействия объектов и включает:

- онтологию домена, которая определяет состав классов объектов домена, их атрибутов и взаимоотношений, а также услуг (операций), которые могут выполнять объекты классов;
- модель поведения задает возможные состояния объектов, инцидентов, инициирующих переходы с одного состояния к другому, а также сообщения, которыми обмениваются объекты;
- модель процессов определяет действия, которые выполняются при проектировании объектов, как компонентов.

Модель требований в UML - это совокупность диаграмм, которые визуализируют основные элементы структуры системы.

Язык моделирования UML поддерживает статические и динамические модели, в том числе модель последовательностей - одну из наиболее полезных и наглядных моделей, в каждом узле которой - взаимодействующие объекты. Все модели представляются диаграммами, краткая характеристика которых дается ниже.

Диаграмма классов (Class diagram) отображает онтологию домена, эквивалентна структуре информационной модели метода С.Шлеера и С.Меллора, определяет состав классов объектов и их взаимоотношений. Диаграмма задается иконами, как визуальное изображение понятий, и связей между ними. Верхняя часть иконы - обязательная, она определяет имя класса. Вторая и третья части иконы определяют соответственно список атрибутов класса и список операций класса.

Атрибутами могут быть типы значений в UML:

- public (общий) обозначает операцию класса, вызываемую из любой части программы любым объектом системы;
- protected (защищенный) обозначает операцию, вызванную объектом того класса, в котором она определена или наследована,
- private (частный) обозначает операцию, вызванную только объектом того класса, в котором она определена.

Пользователь может определять специфические для него атрибуты. Под операцией понимается сервис, который экземпляр класса может выполнять, если к нему будет произведен соответствующий вызов. Операция имеет название и список аргументов.

Классы могут находиться в следующих отношениях или связях.

Ассоциация - взаимная зависимость между объектами разных классов, каждый из которых является равноправным ее членом. Она может обозначать количество экземпляров объектов каждого класса, которые принимают участие в связи (0 - если ни одного, 1 - если один, N - если много).

Зависимость между классами, при которой класс-клиент может использовать определенную операцию другого класса; классы могут быть связаны отношением трассирования, если один класс трансформируется в другой в результате выполнения определенного процесса ЖЦ.

Экземпляризация - зависимость между параметризированным абстрактным классом-шаблоном (template) и реальным классом, который инициирует параметры шаблона (например, контейнерные классы языка C++).

Моделирование поведения системы. *Поведение* системы определяется множеством обменивающихся сообщениями объектов и задается диаграммами: последовательность, сотрудничество, деятельности и состояния.

Диаграмма последовательности применяется для задания взаимодействия объектов, с помощью сценариев, отображающих события, связанные с их созданием и уничтожением. Взаимодействие объектов контролируется событиями, которые происходят в сценарии и поддерживаются сообщениями к другим объектам.

Диаграммы сотрудничества задает поведение совокупности объектов, функции которых ориентированы на достижение целей системы, а также взаимосвязи тех ролей, которые обеспечивают сотрудничество.

Диаграмма деятельности задает поведение системы в виде определенных работ, которые может выполнять система или актер, виды работ могут зависеть от принятия решений в зависимости от заданных условий или ограничений. В качестве примера использования диаграммы деятельности UML приведена структура программы "Оплатить услуги" (рис. 4). Данная диаграмма демонстрирует программу расчета и оплаты услуг. В ней выполняется ряд последовательных действий по расчету стоимости за услуги.

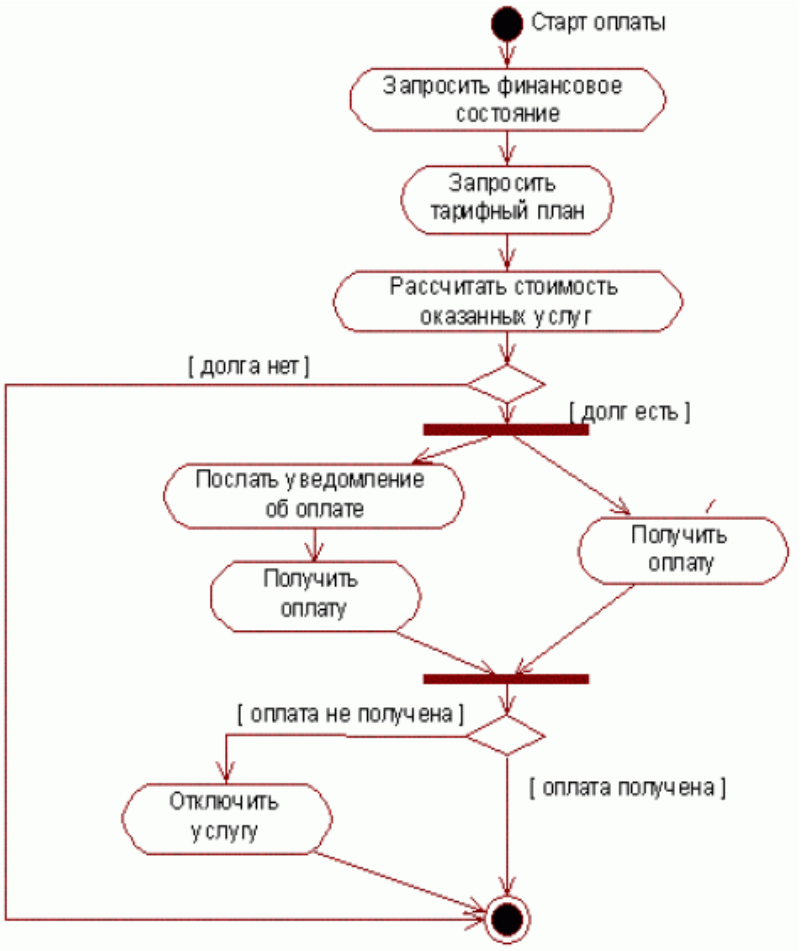


Рисунок 4 - Диаграмма программы расчета и оплаты услуг

В зависимости от выполнения условия "долга нет" происходит переход в конечное состояние или на разделение потоков на два параллельных. В левой ветви выполняется действие "послать уведомление об оплате" и "получить оплату", а в правой - "получить оплату". Распараллеливание означает, что пользователь может оплатить услуги, не дожидаясь уведомления. Параллельные потоки сливаются в один, затем снова ветвление алгоритма - условие "оплата не получена", "отключить услугу" и переход в конечное состояние.

Диаграмма состояний использует расширенную модель конечного автомата и определяет условия переходов, действия при входе и выходе из состояния, а также параллельно действующие состояния. Переход по списку данных инициирует некоторое событие. Состояние зависит от условий перехода, подобно тому, как взаимодействуют две параллельно работающие машины.

Диаграмма реализации состоит из диаграммы компонента и размещения.

Построение ПС методом UML состоит в выполнении этапов ЖЦ, приведенных на общей схема реализации ПрО (рис. 5).

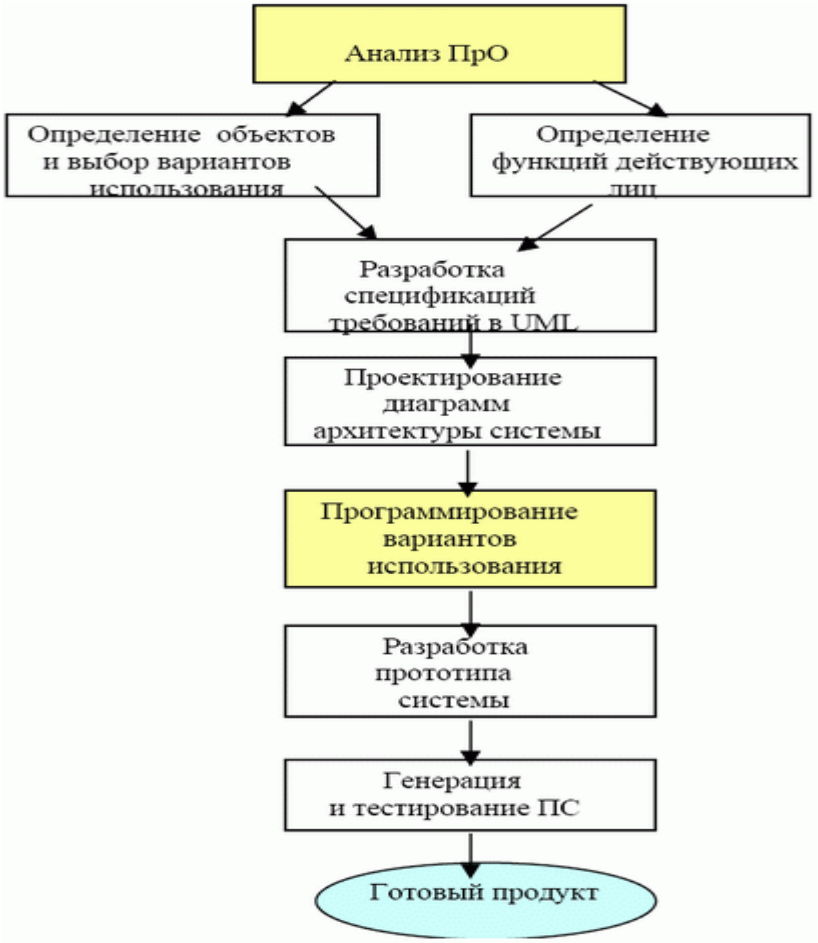


Рисунок 5 - Схема моделирования и проектирования ПС в UML

Диаграмма компонента отображает структуру системы как композицию компонентов и связей между ними. Диаграмма размещения задает состав физических ресурсов системы (узлов системы) и отношений между ними, к которым относятся необходимые аппаратные устройства, на которых располагаются компоненты, взаимодействующие между собой.

Пакет может быть элементом конфигурации построенной системы, на которую можно ссылаться в разных диаграммах.

2.1.4. КОМПОНЕНТНЫЙ ПОДХОД

По оценкам экспертов, 75 % работ по программированию в информационном мире дублируются (например, программы складского учета, начисления зарплаты, расчета затрат на производство продукции и т.п.). Большинство из этих программ типовые, но каждый раз находятся особенности, которые влияют на их повторную разработку.

Компонентное проектирование сложных программ из готовых компонентов является наиболее производительным.

Переход к компонентам происходил эволюционно: от подпрограмм, модулей, функций. При этом усовершенствовались элементы, методы их композиции и накопления для дальнейшего использования (табл. 1).

Компонентный подход дополняет и расширяет существующие подходы в программировании, особенно ООП. Объекты рассматриваются на логическом уровне проектирования ПС, а компоненты - это физическая реализация объектов.

Компоненты конструируются как некоторая абстракция, включающая в себя информационный раздел и артефакты (спецификация, код, контейнер и др.). В этом разделе содержатся сведения: назначение, дата изготовления, условия применения (ОС, среда, платформа и т.п.). Артефакт - это реализация (implementation), интерфейс (interface) и схема развертывания (deployment) компонента.

Реализация - это код, который будет выполняться при обращении к операциям, определенным в интерфейсах компонента. Компонент может иметь несколько реализаций в зависимости от операционной среды, модели данных, СУБД и др. Для описания компонентов, как правило, применяются языки объектно-ориентированной ориентации, а также язык JAVA, в котором понятие интерфейса и класса - базовые, используются в инструментах Javabeans и Enterprise Javabeans и в объектной модели CORBA.

Таблица 5.1. Схема эволюции элементов компонентов

Элемент композиции	**Описание элемента**	**Схема взаимодействия**	**Представление, хранение**	**Результат композиции**
Процедура, подпрограмма, функция	Идентификатор	Непосредственное обращение. оператор вызова	Библиотеки подпрограмм и функций	Программа
Модуль	Паспорт модуля, связи	Вызов модулей. интеграция модулей	Банк, библиотеки модутей	Программа с модульной структурой
Объект	Описание класса	Создание экземпляров классов, вызов методов	Библиотеки классов	Объектно-ориентированная программа
Компонент	Описание логики (бизнес), интерфейсов (APL, IDL), схемы развертывания	Удаленный вызов в компонентных моделях (CQV1 CORBA,OSF,...)	Регозитарий компонентов. серверы и контейнеры компонентов	Распределенное компонентно-ориентированное приложение
Сервис	Описание бизнес-логики интерфейсов сервиса (XML, WSDL, ...)	Удаленный вызов (RPC, НИР, SOAP,...)	Индексация и каталогизация сервисов (XML, UDDL..)	Распределенное сервисо-ориентированное приложение

Интерфейс отображает операции обращения к реализации компонента, описывается в языках IDL или APL, включает в себя описание типов и операции передачи аргументов и результатов для взаимодействия компонентов. Компонент, как физическая сущность, может иметь множество интерфейсов.

Развертывание - это выполнение физического файла в соответствии с конфигурацией (версией), параметрами настройки для запуска на выполнение компонента.

Компоненты наследуются в виде классов и используются в модели, композиции и в каркасе (Фреймворке) интегрированной среды. Управление компонентами проводится на архитектурном, компонентном или интерфейсном уровнях, между которыми существует взаимная связь.

Компонент описывается в языке программирования, не зависит от операционной среды (например, от среды виртуальной машины JAVA) и от реальной платформы (например, от платформ в системе CORBA), где он будет функционировать.

Типы компонентных структур. Расширением понятия компонента является *шаблон* (паттерн) - абстракция, которая содержит описание взаимодействия совокупности объектов в общей кооперативной деятельности, для которой определены роли участников и их ответственности. Шаблон является повторяемой частью программного элемента как схема или взаимосвязь контекста описания для решения проблемы.

Компонентная модель - отражает проектные решения по композиции компонентов, определяет типы шаблонов компонентов и допустимые между ними взаимодействия, а

также является источником формирования файла развертывания ПС в среде функционирования.

Каркас - представляет собой высокоуровневую абстракцию проекта ПС, в которой функции компонентов отделены от задач управления ими. Например, бизнес-логика - это функция компонента, а каркас - управление ими. Каркас объединяет множество взаимодействующих между собою объектов в некоторую интегрированную среду для решения заданной конечной цели. В зависимости от специализации каркас называют "белым или черным ящиком".

Каркас типа "белый ящик" включает абстрактные классы для представления цели объекта и его интерфейса. При реализации эти классы наследуются в конкретные классы с указанием соответствующих методов реализации. Использование такого типа каркаса является характерным для ООП.

Для каркаса типа "черный ящик" в его видимую часть выносятся точки, разрешающие изменять входы и выходы.

Композиция компонентов может быть следующих типов:

- *композиция компонент-компонент* обеспечивает непосредственное взаимодействие компонентов через интерфейс на уровне приложения;
- *композиция каркас-компонент* обеспечивает взаимодействие каркаса с компонентами, при котором каркас управляет ресурсами компонентов и их интерфейсами на системном уровне;
- *композиция компонент-каркас* обеспечивает взаимодействие компонента с каркасом по типу "черного ящика", в видимой части которого находится описание файла для развертывания и выполнения определенной функции на сервисном уровне;
- *композиция каркас-каркас* обеспечивает взаимодействие каркасов, каждый из которых может разворачиваться в гетерогенной среде и разрешать компонентам, входящим в каркас, взаимодействовать через их интерфейсы на сетевом уровне.

Компоненты и их композиции, как правило, запоминаются в репозитарии компонентов, а их интерфейсы в репозитарии интерфейсов.

Повторное использование в компонентном программировании - это применение готовых порций формализованных знаний, добытых во время реализации ПС, в новых разработках.

Повторно используемые компоненты (ПИК) - это готовые компоненты, элементы оформленных знаний (проектные решения, функции, шаблоны и др.) в ходе разработки, которые используются не только самими разработчиками, а и другими пользователями путем адаптации их к условиям новой ПС, что упрощает и сокращает сроки ее разработки. В системе Интернет в данный момент имеется много разных библиотек, репозитариев, содержащих ПИК, и их можно использовать в новых проектах.

При создании компонентов, ориентированных на повторное использование, их интерфейсы должны содержать операции, которые обеспечивают разные способы применения компонентов. Как и любые элементы производства, ПИК должны отвечать определенным требованиям, обладать характерными свойствами и структурой, а также иметь механизмы обращения к ним и др.

Главным преимуществом создания ПС из компонентов является уменьшение затрат на разработку за счет выбора готовых компонентов с подобными функциями, пригодными для практического применения и настройки их к новым условиям, на что тратится меньше усилий, чем на аналогичную разработку.

Поиск готовых компонентов основывается на методах классификации и каталогизации. Метод классификации предназначен для представления информации о компонентах с целью быстрого поиска и отбора. Метод каталогизации - для физического их размещения в репозитариях с обеспечением доступа к ним в процессе интеграции.

Методология компонентной разработки ПС. Создание компонентной системы начинается с анализа ПрО и построения концептуальной модели, на основе которой создается компонентная модель (рис. 7) , включающая проектные решения по композиции

компонентов, использованию разных типов шаблонов, связей между ними и операции развертывания ПС в среде функционирования.

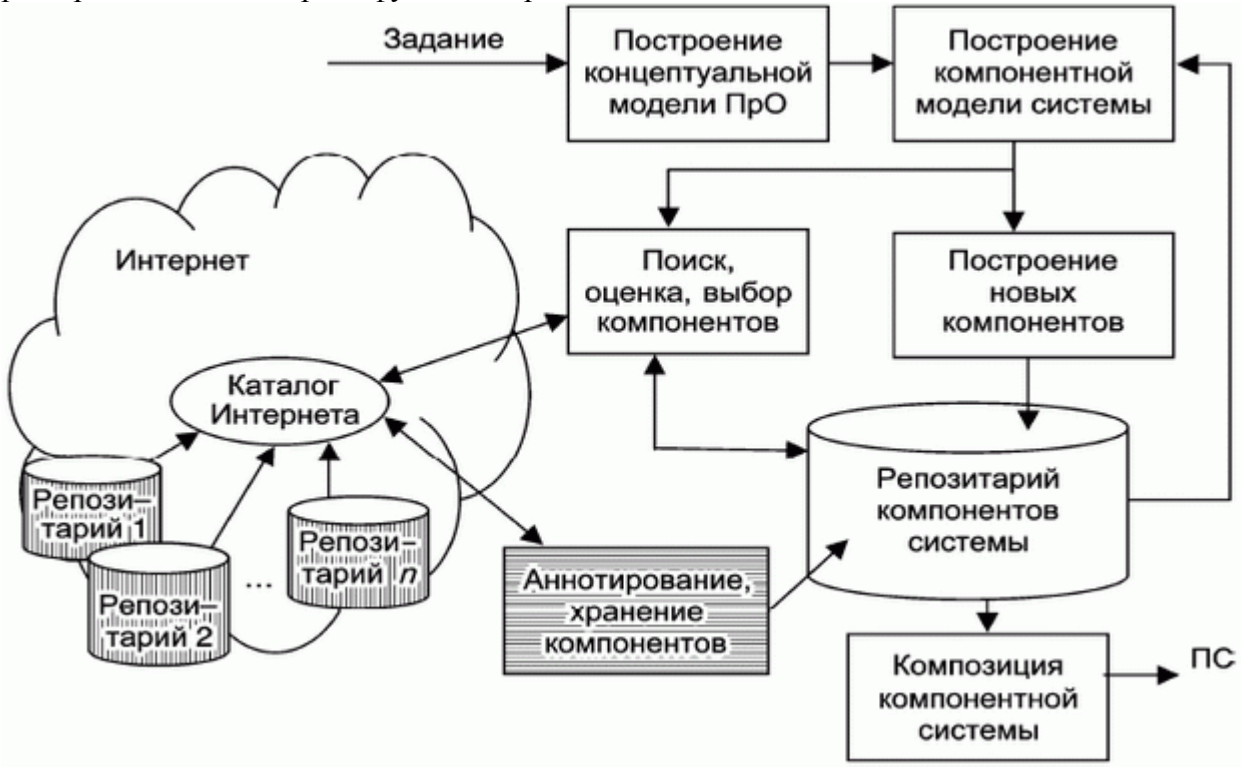


Рисунок 7 - Концептуальна схема построения ПС из компонентов в среде Интернет

Готовые компоненты берутся из репозитариев Интернета и используются при создании ПС, технология построения которых описывается следующими этапами ЖЦ.

1. *Поиск, выбор ПИК* и разработка новых компонентов, исходя из системы классификации компонентов и их каталогизации, формализованное определение спецификаций интерфейсов, поведения и функциональности компонентов, а также их аннотирования и размещения в репозитарии системы или в Интернет.
2. *Разработка требований* (Requirements) к ПС - это формирование и описание функциональных, нефункциональных и др. свойств ПС.
3. *Анализ поведения* (Behavioral Analysis) ПС заключается в определении функций системы, деталей проектирования и методов их выполнения.
4. *Спецификация интерфейсов и взаимодействий компонентов* (Interface and Interaction Specification) отражает распределение ролей компонентов, интерфейсов, их идентификацию и взаимодействие компонентов через поток действий (workflow).

1. 5. *Интеграция набора компонентов и ПИК* (Application Assembly and Component Reuse) в единую среду основывается на подборе и адаптации ПИК, определении совокупности правил, условий интеграции и построении конфигурации каркаса системы.

1. *Тестирование компонентов и среды* (Component Testing) основывается на методах верификации и тестирования для проверки правильности как отдельных компонентов и ПИК, так и интегрированной из компонентов ПС.
2. *Развертывание (System Deployment)* включает оптимизацию плана компонентной конфигурации с учетом среды пользователя, развертку отдельных компонентов и создание целевой компонентной конфигурации для функционирования ПС.
3. *Сопровождения ПС* (System Support and Maintenance) состоит из анализа ошибок и отказов при функционировании ПС, поиска и исправления ошибок, повторного ее тестирования и адаптации новых компонентов к требованиям и условиям интегрированной среды.

2.1.5. АСПЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Аспектно-ориентированное программирование (АОП) - это парадигма построения гибких к изменению ПС путем добавления новых аспектов (функций), обеспечивающих безопасность, взаимодействие компонентов с другой средой, также синхронизацию одновременного доступа частей ПС к данным и вызов новых общесистемных средств.

Аспектом может быть ПИК, фрагмент программы, реализующий концепцию взаимодействия компонентов в среде, защиту данных и др. ПС, которая создается из ПИК, объектов, небольших методов и аспектов, дополняется необходимыми фрагментами взаимодействия, синхронизации, защиты и т.п. путем встраивания их в точки компонентов ПС, где они необходимы. В результате встроенные фрагменты дополняют компоненты новым содержательным аспектом и тем самым значительно усложняют процесс вычислений.

Практическая реализация аспектов, размещенных в разных частях элементов ПС, обеспечивается механизмом перекрестных ссылок и точками соединения, через которые осуществляется связь с аспектным фрагментом для получения определенной дополнительной функции.

В основе АОП лежит метод разбиения задач ПрО на ряд функциональных компонентов, определения необходимости применения разного рода дополнительных аспектов и установления точек расположения аспектов в отдельных компонентах, где это требуется. Эти работы выполняются на этапах ЖЦ процесса разработки, способствуют реализации ПС с ориентацией на взаимодействие компонентов или их синхронизацию. Такой подход известен при проведении отладки программ, когда фрагменты отладочных программ встраиваются в отдельные точки исходной программы для выдачи промежуточных результатов. Когда отладка завершается успешно, эти участки удаляются. В случае аспектов - их программные фрагменты остаются в программе.

Создание конечного продукта ПС в АОП выполняется по технологии, соответствующей разработке компонентных систем, с той особенностью, что используемые аспекты определяют особые условия выполнения компонентов в среде взаимодействия. Аспекты можно рассматривать как выполнение разных ролей взаимодействующими лицами, что приближает аспект к роли программного агента, выполняющего дополнительные функции при определении архитектуры системы и повышение качества компонентов.

Для использования аспектов при выработке проектных решений используется механизм фильтрации входных сообщений, с помощью которых проводится изменение параметров и имен текстов аспектов в конкретно заданном компоненте системы. Код компонента становится "нечистым", когда он пересечен аспектами, и при композиции с другими компонентами общие средства (вызов процедур, RPC, RMI, IDL и др.) становятся недостаточными. Это так как аспекты требуют декларативного сцепления описаний и связано с тем, что фрагменты находятся или берутся из различных объектов. Один из механизмов композиции компонентов и аспектов - фильтр композиции, который обновляет аспекты без изменения функциональных возможностей. Фактически фильтрация касается входных и выходных параметров сообщений, которые переопределяют соответствующие имена объектов. Иными словами, фильтры делегируют внутренним частям компонентов параметры, переадресовывая ранее установленные ссылки, проверяют и размещают в буфере сообщений, локализуют ограничения и готовят компонент для выполнения.

В ОО-программах могут быть методы, выполняющие дополнительно некоторые расчеты с обращением на другие методы внешнего уровня. Деметр сформулировал закон, согласно которому длинные последовательности мелких методов не должны выполняться. В результате создается код алгоритма с именами классов, не задействованных в расчетных операциях, а также дополнительный класс, который расширяет код этими расчетами.

С точки зрения моделирования, аспекты можно рассматривать как каркасы декомпозиции системы, в которых отдельные аспекты пересекают ряд многократно используемых ПИК (рис. 8).

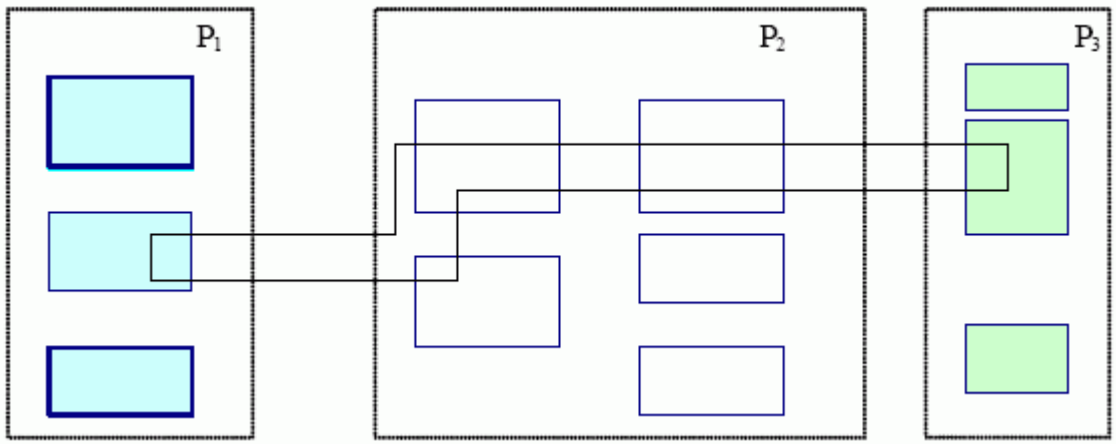


Рисунок 8 - Пример расположения аспектов в программах P1, P2 и P3

Разным аспектам проектируемой системы могут отвечать и разные парадигмы программирования: объектно-ориентированные, структурные и др. Они по отношению к проектируемой ПрО образуют мультипарадигмную концепцию обработки, такую как синхронизация, взаимодействие, обработка ошибок и др. со значительными доработками процессов их реализации. Кроме того, этот механизм позволяет устанавливать аспектные связи с другими предметными областями в терминах родственных областей. Языки АОП позволяют описывать аспекты для разных ПрО. В процессе компиляции пересекаемые аспекты объединяются, оптимизируются, генерируются и выполняются в динамике.

Существенной особенностью АОП является построение модели, которая пересекает структуру другой модели, для которой первая модель является аспектом. Так как аспект связан с моделью, то ее можно перестроить так, чтобы аспект стал, например, модулем и выполнял функцию посредника, реализуя шаблоны взаимодействия. Один из недостатков - пересечение отдельных компонентов аспектами может привести к понижению эффективности их выполнения.

Переплетение аспектов с компонентами проявиться на последующих этапах процесса разработки и поэтому требуется минимизация количества сцеплений между аспектами и компонентами через ссылки в вариантах использования, сопоставление с шаблоном или блоком кода, в котором установлены перекрестные ссылки.

В ходе анализа ПрО и построении ее характеристической модели устанавливается связь с дополнительными аспектами, что приводит к статическому или "жесткому" связыванию компонентов и аспектов модели, учету этого случая при компиляции.

Аспекты с точки зрения моделирования можно рассматривать как каркасы декомпозиции системы с многократным использованием. АОП становится мультипарадигмовой концепцией, сущность которой состоит в том, что разным аспектам проектируемой ПС, должны отвечать разные парадигмы программирования. Каждая из парадигм относительно реализации разных аспектов ПС (синхронизации, внедрения, обработки ошибок и др.) требует их усовершенствования и обобщения для каждой новой ПрО.

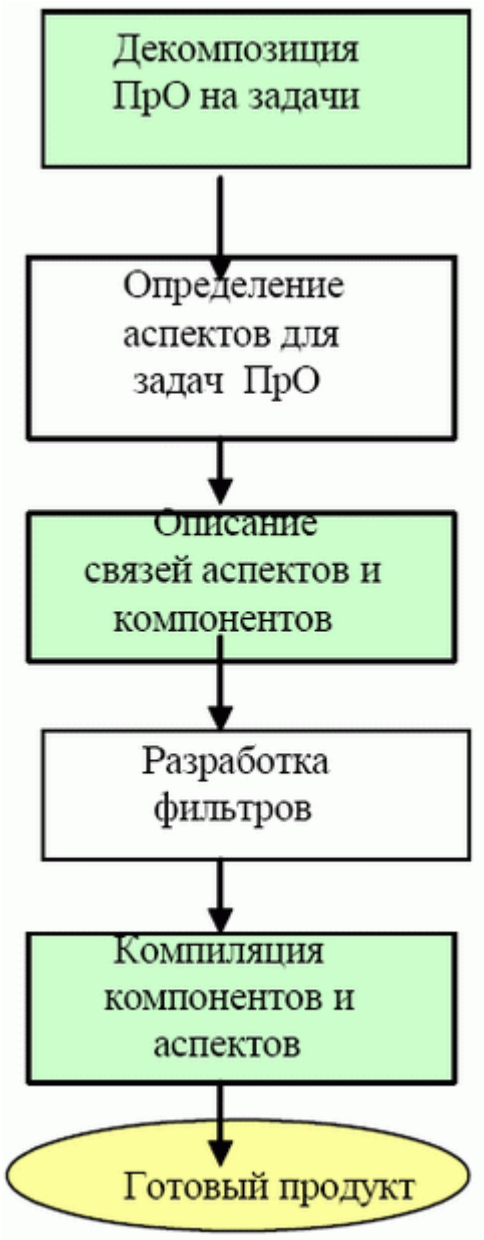


Рисунок 9 - Технологическая схема проектирования ПС средствами АОП

В АОП используется модель модульных расширений в рамках метамодельного программирования, которая обеспечивает оперативное использование новых механизмов композиции отдельных частей ПС или их семейств с учетом предметно-ориентированных возможностей языков (например, SQL) и каркасов, которые поддерживают аспекты. Технология разработки прикладной системы с использованием АОП включает общие этапы (рис. 9):

1. Декомпозиция функциональных задач с условием многоразового применения модулей и выделенных аспектов, т.е. свойств их выполнения (параллельно, синхронно, безопасно и т.д.).
2. Анализ языков спецификации аспектов и определение конкретных аспектов для обеспечения взаимодействия, синхронизации и др. задач ПрО.
3. Определение точек встраивания аспектов в компоненты и формирование ссылок и связей с другими элементами.
4. Разработка фильтров и описание связей аспектов с функциональными компонентами, выделенными в ПрО, отображение фильтров в модели EJB на стороне сервера и управление данными с обеспечением безопасности, защиты доступа к некоторым данным.

5. Определение механизмов композиции (вызовов процедур, методов, сцеплений) функциональных модулей многоразового применения и аспектов в точках их соединения, как фрагментов свойств управления выполнением этих модулей, или ссылок из этих точек на другие модули.

6. Создание объектной или компонентной модели, дополнение ее входными и выходными фильтрами сообщений, посылающих объектам ссылки, задания на выполнение методов или аспектов.

7. Анализ библиотеки расширений для выбора некоторых функциональных модулей, необходимых для реализации задач ПрО.

8. Компиляция, совместная отладка модулей и аспектов, после чего композиция их в готовый программный продукт.

В процессе создания ПС с применением аспектов используются IP-библиотека расширений, активные библиотеки, Smalltalk и ЯП, расширенные средства описания аспектов.

IP-библиотека содержит функции компиляторов, средства оптимизации, редактирования, отображения и др. Например, библиотека матриц для вычисления выражений с массивами, предоставляющая память и др., получила название библиотеки генерирующего типа.

Иной вид библиотек АОП - *активные библиотеки*, которые содержат не только базовый код реализации понятий ПрО, но и целевой код обеспечения оптимизации, адаптации, визуализации и редактирования. Активные библиотеки пополняются средствами и инструментами интеллектуализации агентов, с помощью которых обеспечивается разработка специализированных агентов для реализации конкретных задач ПрО.

2.1.6. ГЕНЕРИРУЮЩЕЕ (ПОРОЖДАЮЩЕЕ) ПРОГРАММИРОВАНИЕ

Генерирующее программирование (generate programming) - генерация семейств приложений из отдельных элементов компонентов, аспектов, сервисов, ПИК, каркасов и т.п. Базис этого программирования - ООП, дополненный механизмами генерации ПИК, другими многоразовыми элементами, а также свойствами их изменчивости, взаимодействия и др..

В нем используются разные методы программирования для поддержки инженерии ПрО как дисциплины инженерного проектирования семейств ПС из разных ранее изготовленных продуктов путем объединения технологии генерации как отдельных ПС, так и их семейств. Эта дисциплина использует методы программирования, соответствующие формализмы и модели для создания более качественных представителей семейства ПС по принципу конвейера.

Главный элемент ПС - это семейство ПС или конкретные его экземпляры, которые генерируются на основе общей генерирующей модели домена (generative domain model) и включающей в себя средства определения членов (представителей) семейства, методы сборки членов семейства и базу конфигурации с набором правил развертывания в операционной среде.

Каждый член семейства создается путем интеграции отдельных компонентов, планирования, контроля и оценки результатов интеграционного тестирования, а также определения затрат на применение многократно используемых ПИК, в том числе из активной библиотеки.

Базовый код элементов активной библиотеки содержит целевой код по обеспечению процедур компиляции, отладки, визуализации и др. Фактически компоненты этих библиотек - это интеллектуальные агенты, генерирующие новых агентов в расширяемой среде программирования для решения конкретных задачи ПрО. Эта среда содержит специальные метапрограммы и компоненты библиотек для осуществления отладки, проверки композиции и взаимодействия компонентов. Среда пополняется

новыми сгенерированными компонентами для членов семейства, в качестве компонентов многоразового применения.

Реализация целей порождающего программирования по включению ПИК в другие члены семейства проводится по двум сформировавшимся инженерным направлениям:

1. *прикладная инженерия* - процесс производства конкретных ПС из ПИК, ранее созданных в среде независимых систем, или как отдельные элементы процесса инженерии некоторой ПрО;
2. *инженерия* ПрО - построение членов семейства или самого семейства систем путем сбора, классификации и фиксации ПИК в качестве их конструктивных элементов, а также для частей систем для конкретной ПрО. Поиск, адаптация ПИК и внедрение их в новые члены семейства ПС проводится с помощью специальных инструментальных средств типа репозитария.

Составная часть инженерии ПрО - инженерия приложений как способ создания отдельных целевых членов семейства для конструирования из этих членов новые ПИК, многократно используемых проектных решений и генерируемых как системы семейства ПрО.

Основные этапы инженерии ПрО приведены на рисунке 10:

- анализ ПрО и выявление объектов и отношений между ними;
- определение области действий объектов ПрО;
- определение общих функциональных и изменяемых характеристик, построение модели, устанавливающей зависимость между различными членами семейства;
- создание базиса для производства конкретных программных членов семейства с механизмами изменчивости независимо от средств их реализации;
- подбор и подготовка компонентов многоразового применения, описание аспектов выполнения задач ПрО;
- генерация отдельного домена, члена семейства и ПС.

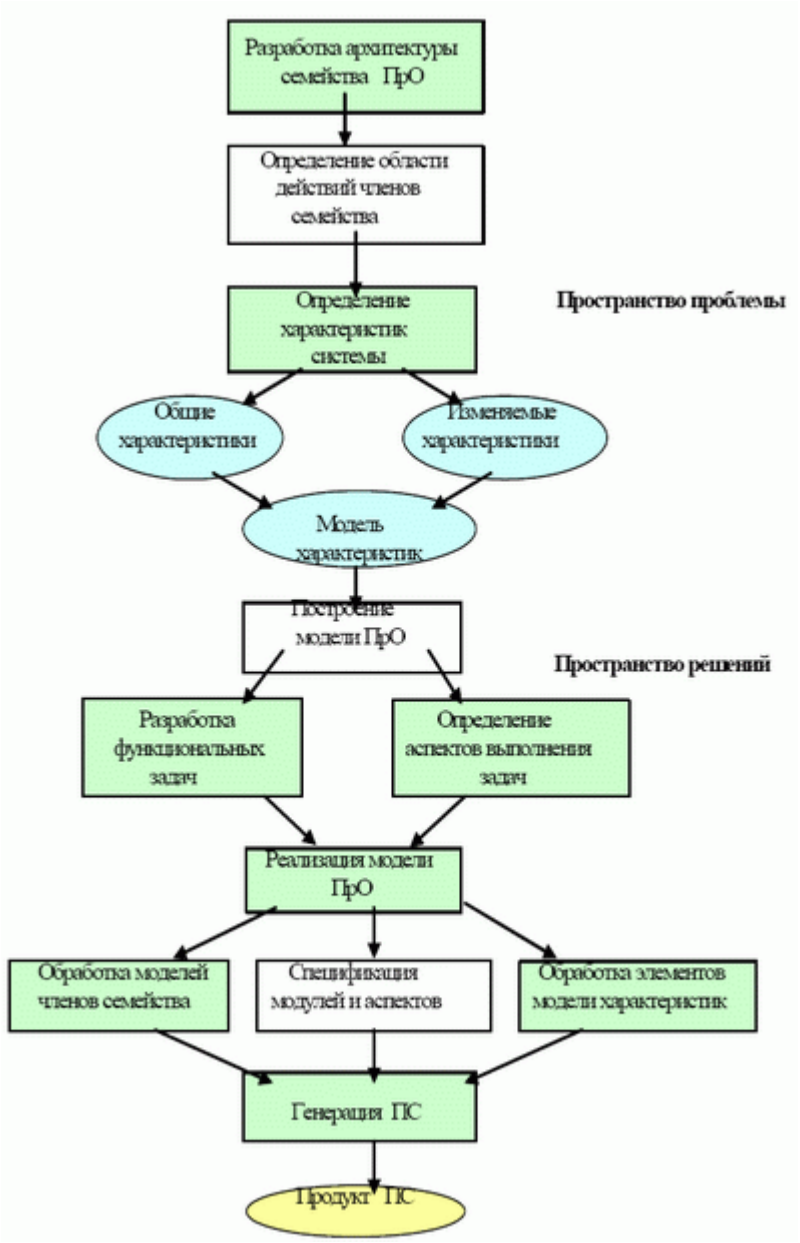


Рисунок 10 - Технологическая схема инженерии ПрО

Генерация доменной модели для семейства ПС основывается на модели характеристик, наборе компонентов реализации задач ПрО, совокупности компонентов и их спецификациях. Результат генерации - готовая подсистема или отдельный член семейства.

К рассмотренной схеме инженерии ПрО также относятся:

- - *корректировка процессов* при включении новых проектных решений или при изменении состава ПИК;
- - *моделирование изменчивости и зависимостей* с помощью механизмов изменения моделей (объектных, взаимодействия и др.), добавления новых требований и понятий, а также фиксации их в модели характеристик и в конфигурации системы;
- - *разработка инфраструктуры ПИК* - описание, хранение, поиск, оценивание и объединение готовых ПИК.

ЖЦ разработки с повторным или многократным использованием обеспечивает получение семейства систем, определение области их действия, а также определение общих и изменяемых характеристик представителей семейства, заданных в модели характеристик. При их определении используются пространство проблемы и пространство решений.

Пространство проблемы (space problem) - компоненты семейства системы, в которых используется ПИК, объекты, аспекты и др., процесс разработки которых включает в себя системные инструменты, а также созданные в ходе разработки ПрО. Инженерия ПрО объединяет в модели характеристик функциональные характеристики, свойства выполнения компонентов, изменяемые параметры разных частей семейства, а также решения, связанные с особенностями взаимодействия групп членов семейства ПС.

Инженерия ПрО обеспечивает не только разработку моделей членов семейства (подсистем), а и моделирование понятий ПрО, модель характеристик для подсистем и набора компонентов, реализующих задачи ПрО. В рамках инженерии ПрО используются горизонтальные и вертикальные типы компонентов в терминологии системы CORBA.

К горизонтальным типам компонентов отнесены общие системные средства, которые нужны разным членам семейства, а именно: графические пользовательские интерфейсы, СУБД, системные программы, библиотеки расчета матриц, контейнеры, каркасы и т.п.

К вертикальным типам компонентов относятся прикладные системы (медицинские, биологические, научные и т.д.), методы инженерии ПрО, а также компоненты горизонтального типа по обслуживанию архитектуры многократного применения компонентов и их интерфейсов и др.

Пространство решений (space solution) - компоненты, каркасы, шаблоны проектирования ПрО, а также средства их соединения или встраивания в ПС и оценки избыточности. Элементы пространства реализуют решение задач этой ПрО. Каркас оснащен механизмом изменения параметров модели, которые требуют избыточную фрагментацию "множество мелких методов и классов". Шаблоны проектирования обеспечивают создание многократно используемых решений в различных типах ПС. Для задания и реализации таких аспектов, как синхронизация, удаленное взаимодействие, защита данных и т.п. применяются технологии ActiveX и JavaBeans, а также новые механизмы композиции и др.

Примером систем поддержки инженерии ПрО и реализации горизонтальных методов является система DEMRAL, предназначенная для разработки библиотек: численного анализа, распознавания речи, графовых вычислений и т.д. Основные виды элементов этой библиотеки - абстрактные типы данных (abstract data types - ADT) и алгоритмы. DEMRAL позволяет моделировать характеристики ПрО и представлять их в характеристической модели и предметно-ориентированных языках описания конфигурации.

Система конструирования RSEB в среде генерирующего программирования использует методы, относящиеся к вертикальным методам, а также ПИК и Use Case при проектировании больших ПС. Методы вертикального типа вызывают различные горизонтальные методы, относящиеся к разным прикладным подсистемам. При работе над отдельной частью семейства могут применяться аспекты взаимодействия, структуры, потоков данных и др. Важную роль при этом выполняет графический пользовательский интерфейс и метод обеспечения взаимодействия компонентов в распределенных средах (например, в CORBA).

2.1.7. АГЕНТНОЕ ПРОГРАММИРОВАНИЕ

Понятие интеллектуального и программного агента появилось более 20 лет назад, их роль в программной инженерии все время возрастает. Так, Джекобсон отметил перспективу использования агентов в качестве менеджеров проектов, разработчиков архитектуры с помощью диаграмм use case и др.

Основной теоретический базис данного программирования - темпоральная, модальная и мультимодельная логики, дедуктивные методы доказательства правильности свойств агентов и др.

С точки зрения программной инженерии, агент это самодостаточная программа, способная управлять своими действиями в информационной среде функционирования для

получения результатов выполнения поставленной задачи и изменения текущего состояния среды. Агент обладает следующими свойствами:

- автономность - это способность действовать без внешнего управляющего воздействия;
- реактивность - это способность реагировать на изменения данных и среды, и воспринимать их;
- активность - это способность ставить цели и выполнять заданные действия для достижения этой цели;
- способность к взаимодействию с другими агентами (или людьми).

Основными задачами программного агента являются:

- самостоятельная работа и контроль своих действий;
- взаимодействие с другими агентами;
- изменение поведения в зависимости от состояния внешней среды;
- выдача достоверной информации о выполнении заданной функции и т.п.

С интеллектуальным агентом связаны знания типа убеждение, намерение, обязательства и т.п. Эти понятия входят в концептуальную модель и связываются между собой операционными планами реализации целей каждого агента. Для достижения целей интеллектуальные агенты взаимодействуют друг с другом, устанавливают связь между собой через сообщения или запросы и выполняют заданные действия или операции в соответствии с имеющимися знаниями.

Агенты могут быть локальными и распределенными (рис. 11). Процессы локальных агентов протекают в клиентских серверах сети, выполняют заданные функции и влияют на общее состояние среды функционирования. Распределенные агенты располагаются в разных узлах сети, выполняют автономно (параллельно, синхронно, асинхронно) предназначенные им функции и могут влиять на общее состояние распределенной среды.

Характер взаимодействия между агентами зависит от совместимости целей, компетентности и т.п.

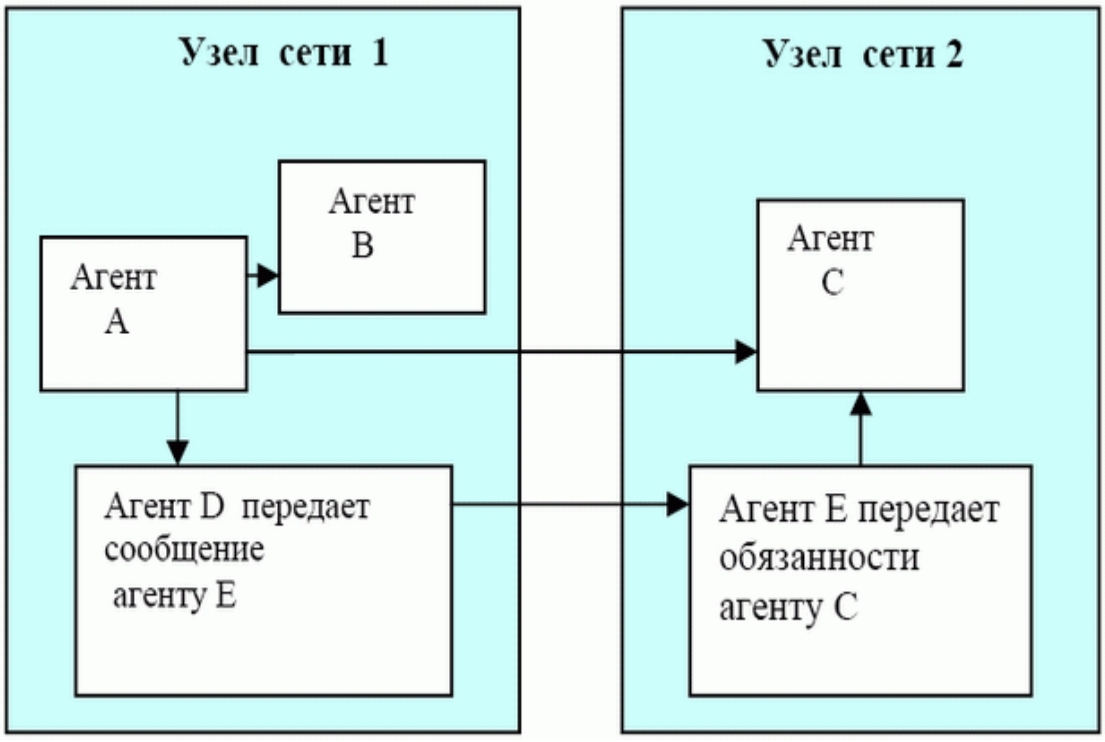


Рисунок 11 - Пример взаимодействия агентов в разных средах

Основу агентно-ориентированного программирования составляют:

- формальный язык описания ментального состояния агентов;
- язык спецификации информационных, временных, мотивационных и функциональных действий агента в среде функционирования;

- язык интерпретации спецификаций агента;
- инструменты конвертирования любых программ в соответствующие агентные программы.

Агенты взаимодействуют между собой с помощью разных механизмов, а именно: координация, коммуникация, кооперация или коалиция.

Под *координацией агентов* понимается процесс обеспечения последовательного функционирования при согласованности их поведения и без взаимных конфликтов. Координация агентов определяется:

- взаимозависимостью целей других агентов-членов коалиции, а также от возможного влияния агентов друг на друга;
- ограничениями, которые принимаются для группы агентов коалиции в рамках общего их функционирования;
- компетенцией - знаниями условий среды функционирования и степени их использования.

Главное средство коммуникации агентов - транспортный протокол TCP/IP или протокол агентов ACL (Agent Communication Languages). Управление агентами (Agent Management) выполняется с помощью сервисов: передача сообщений между агентами, доступ агента к серверу и т.п. *Коммуникация* агентов базируется на общем протоколе, языке HTML и декларативном или процедурном (Java, Telescript, ACL и т.п.) языке описания этого протокола.

Примером активной и скоординированной деятельности агентов по поиску необходимой информации является среда Интернет. В нем агенты обеспечивают доступ к информационным ресурсам, а также выполняют ее анализ, интеграцию, фильтрацию и передачу результата запроса пользователю.

Каждый агент выполняет определенную функцию, передает друг другу задание на последующее действие по доступу к информационному ресурсу, извлечению необходимой информации и передачи ее для обработки следующим агентам. При этом могут возникать нерегулярные состояния (тупики, отсутствие ресурса и др.).

Одной из систем построения агентов, основанной на обмене сообщениями в ACL, является JATLite, которая с помощью Java-классов создает новых агентов, вычисляющих определенные функции в распределенной среде. Система Agent Builder предназначена для конструирования программных агентов, которые описываются в языке Java и могут взаимодействовать на языке KQML (Knowledge Guery and Manipulation Language).

Построенные агенты выполняют функции: менеджера проекта и онтологий, визуализации, отладки и др. Реализацию механизмов взаимодействий агентов обеспечивает система JAFMAS, ряд других мультиагентных систем.

2.2. Теоретическое программирование

Теоретическое программирование включает в себя формальные методы, основанные на спецификации программ, и методы, основанные на математических дисциплинах (логика, алгебра, комбинаторика) и обеспечивающие математический метод анализа и осмысления задач ПрО, а также разработку программ с математической символикой, правильность которых надо доказывать, чтобы получить на компьютере требуемые результаты. В связи с этим получили развитие отдельные направления в теории программирования, которые и предоставляют аппарат конструирования структур программ с применением математических методов.

Так, направление, связанное с использованием математического аппарата для описания действий над объектами как алгебраические операции базовых основ алгебры, получило название алгебраического программирования, алгоритмики и др. Украинские ученые разработали теоретические направления программирования, в частности:

- алгебраическое, инсерционное программирование (Летичевский А.А. и др.);

- экспликативное программирование (Редько В.Н.) и наука о программах (программология), объединяющая логический и математический аппарат для конструирования программ;
- алгебро-алгоритмическое программирование (Цейтлин Г.Е.), объединяющее алгебраический аппарат, теорию алгоритмов и операции над множествами. Остановимся на их краткой характеристике.

2.2.1. АЛГЕБРАИЧЕСКОЕ, ИНСЕРЦИОННОЕ ПРОГРАММИРОВАНИЕ

Алгебраическое программирование (АП) - это конструирование программ с алгебраическими преобразованиями и функциями интеллектуальных агентов. В основе математического аппарата АП лежит алгебра языка действий AL (Action Language) и понятие транзитивной системы, в качестве механизма определения поведения систем и механизмов ее эквивалентности. В качестве понятий в общем случае могут быть компоненты, программы и их спецификации, объекты, взаимодействующие друг с другом и со средой их существования.

Основу АП составляет математическая модель, которая включает в себя следующие понятия:

- агент как транзитивная система с поведением и его завершением;
- поведение агентов, задаваемое в языке AL с помощью операций $a, u, u + v$, констант Δ, $\perp$, 0, граничных условий и рекурсий;
- среда, состоящая из агентов и функций погружения, которая обозначается env и имеет в качестве параметров состояние среды и агентное выражение;
- функция развертывания функциональных выражений в простые агентные выражения;
- транзитивная система, представленная в виде композиции среды и системы взаимодействующих агентов, погруженных в эту среду.

Всякая транзитивная система имеет историю функционирования, которая включает последовательность действий и состояний -конечных или бесконечных. История функционирования хранит одно из соответствующих состояний: успешное завершение вычислений в среде транзитивной системы; тупиковое состояние, когда каждая из параллельно выполняющихся частей системы находятся в состоянии ожидания; неопределенное состояние, возникающее при выполнении алгоритма с бесконечными циклами.

Расширение понятия транзитивных систем - это множество заключительных состояний с успешным завершением функционирования системы и без неопределенных состояний. Главный инвариант состояния транзитивной системы - поведение системы, задаваемое выражениями алгебры поведения $F(A)$ на множестве операций алгебры действий - префиксинг $a \cdot u$, недетерминированный выбор $u + v$ одного из двух поведений u и v со свойством ассоциативности и коммутативности. Конечное поведение системы задается константами Δ, $\perp$, 0, которые обозначают соответственно состояние успешного завершения, неопределенного и тупикового. Алгебра поведения включает отношение $\leq$, элемент $\perp$ как наименьший и операции поведения, являющиеся монотонными. Средствами алгебры поведения $F(A)$ доказана теорема про наименьшую неподвижную точку.

Транзитивные системы называют бисимуляцийно эквивалентными, если каждое состояние эквивалентно состоянию другой системы. На множестве поведений определяются новые операции, которые используются для построения программ агентов. К ним относятся следующие операции: последовательная композиция ($u; v$) и параллельная композиция $u \| v$.

Среда E, где находится объект, определяется как агент в алгебре действий AL и функции погружения от двух аргументов $Ins(e,u) = e[u]$. Первый аргумент - это поведение среды, второй - поведение агента, который погружается в эту среду с заданным состоянием. Значения функций погружения - это новое состояние одной и той же среды. Базовым понятием является "действие", которое трансформирует состояние агентов, поведение которых, в конце концов, изменяется.

Поведение агентов характеризуется состоянием с точностью до слабой эквивалентности. Каждый агент рассматривается как транзитивная система с действиями, определяющими не детерминированный выбор и последовательную композицию (т.е. примитивные и сложные действия). Последовательная композиция - ассоциативная, а параллельная композиция - ассоциативная и коммутативная. Параллельная композиция раскладывается на комбинацию действий компонентов.

Взаимодействие агентов может быть двух типов. Первый тип выражается через параллельную композицию агентов над той же самой областью действий и соответствующей комбинацией действий. Другой тип выражается через функцию погружения агента в некоторую среду, результат трансформации - новая среда.

Язык действий AL имеет синтаксис и семантику. Синтаксис языка задает правила описания действий, семантика - функции, которые определяются средствами и выражениями языка и ставят в соответствие заданным выражениям значения в некоторой семантической области. Разные семантические функции могут давать равные абстракции и свойства программ. Семантика может быть вычислительной и интерактивной. Каждая алгебра действий - это гомоморфный образ алгебры примитивных действий, когда все слагаемые разные, а их представление однозначно с точностью до ассоциативности и коммутативности при детерминированном выборе.

Агенты рассматриваются как значения транзитивных систем с точностью до бисимиляционной эквивалентности, которая характеризуется непрерывной алгеброй с аппроксимацией и двумя операциями: недетерминированным выбором и префиксингом. Среда вводится как агент, в нее погружается функция, которая имеет поведение (агент и среда). Произвольные непрерывные функции могут быть использованы как функции погружения в эти функции. Трансформации поведения среды, которые определяются функциями погружения, составляют новый тип эквивалентности - эквивалентность погружения.

Создание новых методов программирования с введением агентов и сред позволяет интерпретировать элементы сложных программ как самостоятельно взаимодействующие объекты.

В АП интегрируется процедурное, функциональное и логическое программирование, используются специальные структуры данных - граф термов, который разрешает использовать разные средства представления данных и знаний о ПрО в виде выражений многоосновной алгебры данных.

Наибольшую актуальность имеют системы символьных вычислений, которые дают возможность работать с математическими объектами сложной иерархической структуры - группы, кольца, поля. Теория АП обеспечивает создание математической информационной среды с универсальными математическими конструкциями, вычислительными механизмами, учитывающими особенности разработки ПС и функционирования.

Алгебраическое программирование концентрирует внимание на проблемах интеллектуализации и аспектах поведения агентов в распределенной среде, куда они погружаются. Оно постепенно перешло в *инсерционное программирование* путем вставки, погружения агентов в разнообразные среды для преобразования поведения агентов на основе модели поведения, соответствующей размеченной транзитивной системы и бисимуляционной эквивалентности. Данное программирование обобщает алгебраическое

преобразование множества состояний информационной среды на объекты, обладающие поведением. Схема создания агентных программ представлена на рисунке 12.

Главное действующее лицо инсерционного программирования - агент, обладающий поведением при его погружении в среду, которую он также меняет. Характерной особенностью является недетерминированное поведение агентов и сред, как это происходит в реальных системах. При этом программа агента требует не интерпретации, а моделирования, поскольку она представляется транзитивной системой в виде композиции среды и системы взаимодействующих агентов, погруженных в эту среду. Язык действий - порождающий, в нем задается синтаксис действий, агентных функциональных выражений и внутренних состояний среды. Кроме того, в этом языке описывается семантика функций развертывания агентных функциональных выражений и погружений агентов в среду.

Программа - набор параметров и начальных агентных выражений. Параметры могут быть фиксированные и переменные, изменяющиеся при переходе от одной среды в другую. Особенно это касается параметров развертывания.

Рисунок 12 - Технологическая схема в АП

На следующем уровне описания программы находятся функции, которые преобразуют любое агентное выражение в AL-программу действий. Например, функция

U развертывания функциональных выражений может быть представлена в виде простых агентных выражений:

$$U(u+v) = U(u) + U(v), U(u) = u,$$

где u, v - параметры агентных выражений.

Функция развертывания агентных функциональных выражений задает семантику AL-программы в виде программы функционирования агента в любой среде. Если функция развертывания обрабатывается за конечное число шагов, то она содержит конечное выражение. Множество всех переходов, заключительных и тупиковых состояний - перечислимое, даже если функция развертывания имеет бесконечное множество нетривиальных итераций. Состоянием транзитивной системы являются ограниченные выражения, определяемые операцией выбора $+$, соотношением $x + 0 = a$ и отношением перехода с правилом $u \to U(u)$.

Третий уровень программы - это функция развертывания погружений, задаваемых в алгебре действий. Эти функции определяются на объектах среды и агентов, т.е. агентных выражениях. Для корректности функций погружения необходимо, чтобы она зависела только от поведения агента и среды и была непрерывной функцией. Следующим шагом разработки программ является ее реализация в ЯП, например в C++, когда уточняются типы данных и параметры. Затем проводится верификация полученной программы для проверки правильности выполнения ее поведения в заданной модели.

2.2.2. ЭКСПЛИКАТИВНОЕ, НОМИНАТИВНОЕ ПРОГРАММИРОВАНИЕ

Экспликативное программирование (ЭП) ориентировано на разработку теории дескриптивных и декларативных программных формализмов, адекватных моделям структур данных, программ и средств конструирования из них программ. Для этих структур решены проблемы существования, единства и эффективности. Теоретическую основу ЭП составляют логика, конструктивная математика, информатика, композиционное программирование и классическая теории алгоритмов. Для изображения алгоритмов программ используются алгоритмические языки и методы программирования: функциональное, логическое, структурное, денотационное и др.

Принципами ЭП являются:

- *принцип развития* понятия программы в абстрактном представлении и постепенной ее конкретизации с помощью экспликаций;
- *принцип прагматичности* или полезности определения понятия программы выполняется с точки зрения понятия "проблема" и ориентирован на решение задач пользователя;
- *принцип адекватности* ориентирован на абстрактное построение программ и реализацию проблемы с учетом *информационности данных* и *апликативности*. Программа рассматривается как функция, вырабатывающая выходные данные на основе входных данных. Функция - это объект, которому сопоставляется *денотат* имени функции с помощью отношения *именования* (*номинации*) *;*
- *принцип дескриптивности* позволяет трактовать программу как сложные дескрипции, построенные из более простых и композиций отображения входных данных в результаты на основе п *ринципа вычислимости*.

Развитие понятия функции осуществляется с помощью *принципа композиционности*, т.е. составления программ (функций) из более простых программ для создания новых объектов с более сложными именами (дескрипциями) для функций. Они включают номинативные (именные) и языковые выражения, термы и формулы.

Таким образом, процесс развития программы осуществляется в виде цепочки понятий:

данные - функция - имя функции - композиция - дескрипция.

Из них триада - "данные - функция - композиция" задает семантический аспект программы, а триада "данные - имя функции - дескрипция" - синтаксический аспект.

Главным в ЭП является семантический аспект, система композиций и номинативности (КНС), ориентированная на систематическое описание номинативных отношений при построении данных, функций, композиций и дескрипций.

КНС содержит специальные языковые системы для описания разнообразных классов функций, которые называются композиционно-номинативными языками функций. Такие системы тесно связаны с алгебрами функций и данных, построены в семантико- синтаксическом стиле. Они отличается от традиционных систем (моделей программ) теоретико-функциональным подходом, классами однозначных n -арных функций номинативными отображениями и структурами данных.

Для построения математически простых и адекватных моделей программ параметрического типа используется КНС и методы универсальной алгебры, математической логики и теория алгоритмов. Данные в КНС рассматриваются на трех уровнях: абстрактном, булевском и номинативном. Класс номинативных данных обеспечивает построение именных данных, многозначных номинативных данных или мультииименных данных, задаваемых рекурсивно.

В рамках ЭП разработаны новые средства для определения систем данных, функций и композиций номинативного типа, имена аргументов которых принадлежат некоторому множеству имен Z, т.е. композиция определяется на *Z-номинативных* наборах именных функций.

Номинативные данные позволяют задавать структуры данных, которым присущи неоднозначность именования компонентов типа множества, мультимножества, реляции и т.п.

Функции обладают свойством апликативности, их абстракции задают соответственно классы слабых и сильных аппликативных функций. Слабые функции позволяют задавать вычисление значений на множестве входных данных, а сильные - обеспечивают вычисление функций на заданных данных.

Композиции классифицируются уровнями данных и функций, а также типами аргументов. Экспликация композиций соответствует абстрактному рассмотрению функций как слабо аппликативных функций, и их уточнение строится на основе понятия детерминанта композиции как отображение специального типа. Класс аппликативных композиций предназначен для конструирования широкого класса программ.

Практическая проверка теоретического аппарата формализации дедуктивных и ОО БД прошла в ряде экспериментальных проектов в *классе манипуляционных* данных БД заданных в SQL-подобных языках.

2.2.3. АЛГОРИТМИКА ПРОГРАММ

Алгоритмика программ - структурная схемотология построения последовательных и параллельных программ с аппаратом формальных алгебраических преобразований и канонических форм описания логических и операторных выражений. Ее основу составляют системы алгебраических алгебр (САА), расширенные формализмами для представления логических условий параллельных программ и методами символьной мультиобработки.

Построение и исследование алгебры алгоритмов началось с проектирования логических структур ЭВМ под руководством академика В.М.Глушкова. В результате была построена теория САА, которая затем Г.Е.Цейтлиным была положена в основу создания обобщенной теории структурированных схем алгоритмов и программ, называемой *алгоритмикой*.

Основными понятиями алгебры алгоритмики являются:

- операции над множествами, булевы операции, предикаты, функции и операторы;
- бинарные и n-арные отношения, эквивалентность, частично и полностью упорядоченные множества;
- граф-схемы и операции над графовыми структурами;

- операции сигнатуры САА, аксиомы и правила вывода свойств программ на основе сверточной, разверточной и комбинированной стратегий;
- символьная обработка и методы синтаксического анализа программ.

Объекты алгоритмики - модели алгоритмов и программ, представляемые в виде схем. Метод алгоритмики базируется на компьютерной алгебре и логике и используется для проектирования алгоритмов прикладных задач. Построенные алгоритмы описываются в ЯП и реализуются соответствующими системами программирования в машинное представление.

В рамках алгоритмики разработаны специальные инструментальные средства реализации алгоритмов программ, которые используют современные объектно-ориентированные средства и метод моделирования UML. Тем самым обеспечивается полный цикл работ по практическому применению разработанной теории алгоритмики для реализации прикладных задач, начиная с постановки задачи, формирования требований и разработки алгоритмов до получения программ решения этих задач (рис. 13).

Алгебра алгоритмов. Под алгеброй алгоритмов $\mathrm{AA} = \{A, \Omega\}$ понимается основа Aи сигнатура Ωопераций над элементами основы алгебры. С помощью операции сигнатуры может быть получен произвольный элемент $q \in AA$, который называется системой образующих алгебры.

Если из этой системы не может быть исключен ни один элемент без нарушения ее свойств, то такая система образующих называется базисом алгебры.

Операции алгебры удовлетворяют следующим аксиоматическим законам: ассоциативности, коммутативности, идемпотентности, закону исключения третьего и противоречия. Алгебра, которой удовлетворяют перечисленные операции, называется булевой.

В алгебре алгоритмов используется алгебра множеств, элементами которой являются сами множества и операции над ними (объединение, пересечение, дополнение, универсум и др.).

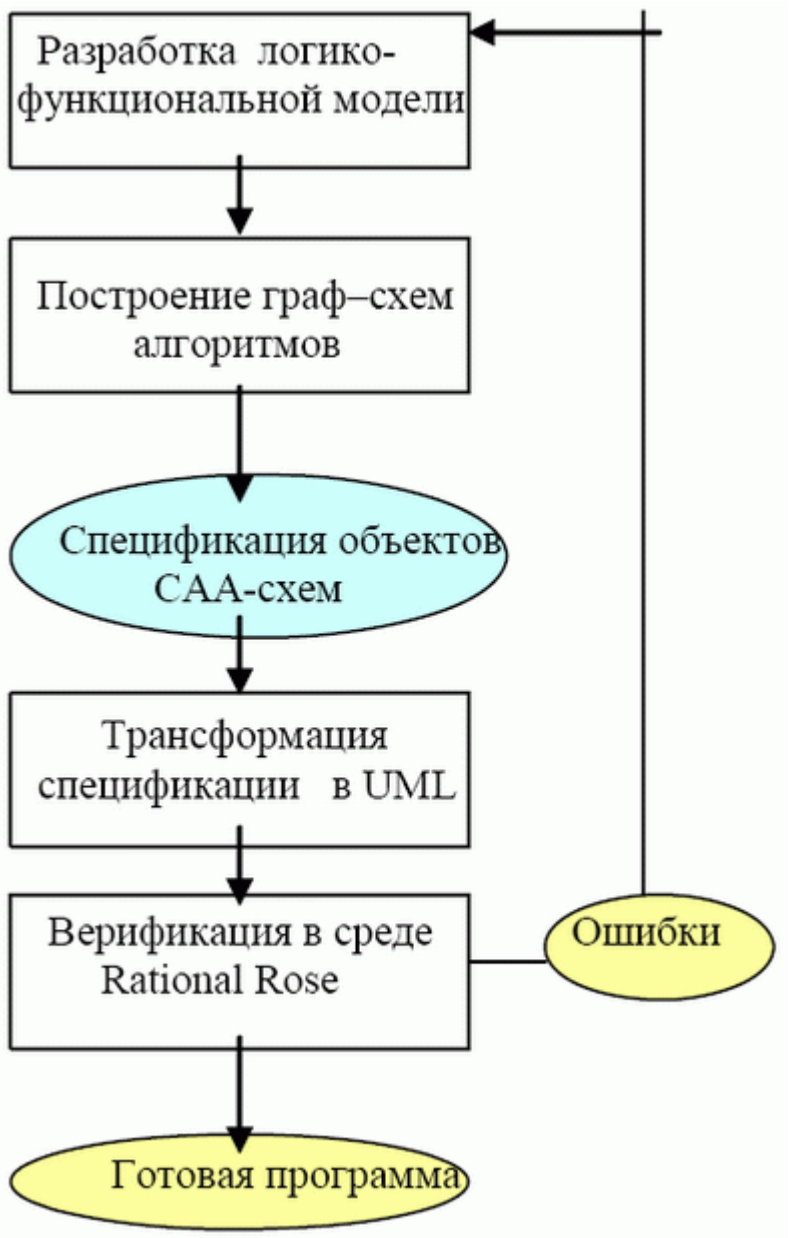


Рисунок 13 - Схема проектирования программ в алгебре алгоритмики

Основные объекты алгебры алгоритимики - схемы алгоритмов и их суперпозиции, т.е. подстановки одних схем в другие. С подстановкой связана развертка, которая соответствует нисходящему процессу проектирования алгоритмов, и свертка. Т.е. переход к более высокому уровню спецификации алгоритма. Схемы алгоритмов соответствуют конструкциям структурного программирования.

Последовательное выполнение операторов A и B записывается в виде композиции $A * B$; альтернативное выполнение операторов A и B ($u(A, B)$) означает, если u истинно, то выполняется A,иначе B; цикл $(u(A, B))$ выполняется, пока не станет истинным условие u (u -логическая переменная).

С помощью этих элементарных конструкций строится более сложная схема Π алгоритма:

$$\Pi ::= \{[u_1]\mathrm{A}_1\},$$
$$\mathrm{A}_1 ::= \{[u_2]\mathrm{A}_2 * D\}$$
$$\mathrm{A}_2 ::= \mathrm{A}_3 * C,$$
$$\mathrm{A}_3 ::= \{[u]\mathrm{A}, B\},$$
$$u ::= u_2 \wedge u_1.$$

Проведя суперпозицию путем свертки данной схемы алгоритма Π, получаем формулу:

$$\Pi ::= \{[u_1]\{[u_2]([u_2 \wedge u_1]\mathrm{A}, \mathrm{B}) * \mathrm{C}\} * D\}.$$

Важный результат в области алгоритмики - проведение сопоставительного анализа аппарата алгебры алгоритмики с известными алгебрами. Дадим краткую характеристику.

Алгебра Дейкстры АД = $\{\mathrm{ACC}, L(2), \mathrm{СИГН}\}$ - двухосновная алгебра, элементами которой являются множество ACCоператоров, представленных структурными блок-схемами, множество $L(2)$булевых функций в сигнатуре $\mathrm{СИГН}$, в которую входят операции дизъюнкции, конъюнкции и отрицания, принимающие значения из $L(2)$. С помощью специально разработанных механизмов преобразования АД в алгебру алгоритмики установлена связь между альтернативой и циклом, т.е. $\{[u]\mathrm{A}\} = ([u]\mathrm{E}, \mathrm{A} * \{[u]\mathrm{A}\})$, произвольные операторы представлены суперпозицией основных операций и констант.

Операция фильтрации $\Phi(u) = \{[u]\mathrm{E}, N\}$ в АД представлена суперпозицией тождественного Eи неопределенного N операторов и альтернативы алгебры алгоритмики, где N - фильтр разрешения выполнения операций вычислений.

Оператор цикла $while\ do$ также представлен суперпозицией операций композиции и цикла в алгебре алгоритмики.

Алгебра схем Янова $\mathrm{АЯ} =< \{AHC, L2)\}; СИГН >$, где $\mathrm{АНС}$-совокупность неструктурных схем, $L(2)$- совокупность различных булевских функций, $\mathrm{СИГН}$- сигнатура из композиции $\mathrm{A} * \mathrm{B}$и операция неструктурного перехода $\Pi(u, F)$, а также операции дизъюнкции, конъюнкции и отрицания. $\mathrm{АЯ}$включает операции построения неструктурных логических схем программ. Схема Янова состоит из предикатных символов множества $P(p_1, p_2, \dots,)$, операторных символов множества $A\{\mathrm{a}_1 a_2, \dots\}$ и графа переходов. Оператор в данной алгебре - это пара $A\{p\}$, состоящая из символов множества Aи множества предикатных символов. Граф перехода представляет собой ориентированный граф, в вершинах которого располагаются преобразователи, распознаватели и один оператор останова. Дуги графа задаются стрелками и помечаются они знаками $+$ и $-$. Преобразователь имеет один преемник, а распознаватель - два. Каждый распознаватель включает в себя условие выполнения схемы. Преобразователь обрабатывает операторы, включающие логические переменные, принадлежащие множеству $(p_1, p_2, \dots)$.

Каждая созданная схема $\mathrm{АЯ}$отличается большой сложностью, требует серьезного преобразования при переходе к представлению программы в виде соответствующей последовательности действий, условий перехода и безусловного перехода. В работе [5.30] разработана теория интерпретации схем Янова и доказательство эквивалентности двух операторных схем исходя из особенностей алгебры алгоритмики.

Для представления схемы Янова аппаратом алгебры алгоритмики сигнатура операций $\mathrm{АЯ}$вводятся композиции $\mathrm{A} * \mathrm{B}$и операции условного перехода, который в зависимости от условия u выполняет переход к следующим операторам или к оператору, помеченному меткой (типа $goto$). Условный переход трактуется как бинарная операция

$\Pi(u, F)$, которая зависит от условия u и разметок схемы F. Кроме того, производится замена альтернативы и цикла типа $while\ do$. В результате выполнения бинарных операций получается новая схема F, в которой установлена $\Pi(u)$ вместо метки и булевы операции конъюнкции и отрицания. Эквивалентность выполненных операций преобразования обеспечивает правильность неструктурного представления.

Система алгебр Глушкова $АГ = \{ОП, УС, СИГН\}$, где $ОП$ и $УС$- множества операторов суперпозиции, входящих в сигнатуру $СИГН$, и логических условий, определенных на информационном множестве $ИМ$, $СИГН = \{СИГНад \cup Прогн.\}$, где $СИГНад$- сигнатура операций Дейкстры, $Прогн$. - операция прогнозирования. Сигнатура САА включает в себя операции алгебры АД, обобщенной трехзначной булевой операции и прогнозирования (левое умножение условия на оператор $u = (A * u')$ с порождением предиката $u = УС$ такого, что $u(m) = u'(m'),\ m' = A(m), A \in ОП)$. ИМ - множество обрабатываемых данных и определение операций из множеств ОП и УС. Сущность операция прогнозирования состоит в проверке условия u в состоянии m оператора A и определения условия u', вычисленного в состоянии m' после выполнения оператора A. Данная алгебра ориентирована на аналитическую форму представления алгоритмов и оптимизацию алгоритмов по выбранным критериям.

Алгебра булевых функций и связанные с ней теоремы о функциональной полноте и проблемы минимизации булевых функций также сведены до алгебры алгоритмики.
Алгебра алгоритмики и прикладные подалгебры. Алгебра алгоритимки пополнена двухуровневой алгебраической системой и механизмами абстрактного описания данных (классами алгоритмов). Под многоосновной алгоритмической системой (МАС) понимается система $S = \{\{D_i | i \in I\}; СИГН_0, СИГН_n\}$, где D_i - основы или сорта, $СИГН_0, СИГН_n$ - совокупности операций и предикатов, определенных на D_i. Если они пусты, то определяются многоосновные модели - алгебры. Если сорта интерпретируются как множество обрабатываемых данных, то МАС представляет собой концепцию АТД, в виде подалгебры, широко используемую в объектном программировании. Тем самым устанавливается связь с современными тенденциями развития современного программирования.

Практическим результатом исследований алгебры алгоритмики является построение оригинальных инструментальных систем проектирования алгоритмов и программ на основе современных средств поддержки ООП (Rational Rose). Детальное знакомство с данной темой можно продолжить, изучая приведенные ниже источники.

Контрольные вопросы и задания

1. Дайте характеристику структурного метода программирования.
2. Приведите основные особенности и возможности объектно-ориентированного программирования.
3. Какие диаграммы имеются в языке UML для визуального проектирования программ?
4. Приведите основные типы компонентов и пути их развития в компонентном программировании.
5. Назовите базовые понятия в компонентном программировании.
6. Приведите базовые структуры в компонентном программировании.
7. Определите основные понятия и этапы жизненного цикла компонентного программирования.
8. Определите основные элементы аспектно-ориентированного программирования.
9. Определите основные элементы агентного программирования.

10. Определите понятие агента и его место в программировании.
11. Дайте характеристику инженерии ПрО.
12. Определите объекты генерирующего программирования и дайте краткую характеристику.
13. Что такое пространство проблем и пространство решений?
14. Представьте теоретические методы программирования.
15. Что такое алгоритмика и ее алгебра?
16. Покажите сущность перехода от других алгебр к алгебре алгоритмики.

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6.1.1. Основная литература				
Авторы, составители		Заглавие	Издательство, год	Адрес
Никифоров С. Н.		Прикладное программирование: учебное пособие	, 2018	https://e.lanbook.com/book/106735
Ковалевская, Е. В., Комлева, Н. В.		Методы программирования: учебное пособие	Москва: Евразийский открытый институт, 2011	http://www.iprbookshop.ru/10784.html
Малявко, А. А.		Формальные языки и компиляторы: учебник	Новосибирск: Новосибирский государственный технический университет, 2014	http://www.iprbookshop.ru/47725.html
6.1.2. Дополнительная литература				
Авторы, составители		Заглавие	Издательство, год	Адрес
Петров, В. Ю.		Информатика. Алгоритмизация и программирование. Часть 1: учебное пособие	Санкт-Петербург: Университет ИТМО, 2016	http://www.iprbookshop.ru/66473.html
Влацкая, И. В., Заельская, Н. А., Надточий, Н. С.		Проектирование и реализация прикладного программного обеспечения: учебное пособие	Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2015	http://www.iprbookshop.ru/54145.html
6.1.3. Методические разработки				
Авторы, составители		Заглавие	Издательство, год	Адрес
Л3.1	Глухов М. М., Козлитин О. А., Шапошников В. А., Шишков А. Б.	Задачи и упражнения по математической логике, дискретным функциям и теории алгоритмов	, 2008	http://e.lanbook.com/books/element.php?pl1_cid=25&pl1_id=112

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Информационное обеспечение стратегического планирования» для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Информационное обеспечение стратегического планирования» содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии, направленность (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины – дать представление о перспективах развития методов осуществления информационного обеспечения стратегического планирования, изучить цели, задачи, методы и способы осуществления информационного обеспечения, сформировать умения использовать методы информационного обеспечения на практике.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

В результате освоения данной дисциплины формируется следующая компетенция у обучающегося:

ПК-3.1: Адаптирует бизнес-процессы заказчика к возможностям информационной системы

Изучив данный курс, студент должен:

Знать:

принципы системного представления основных этапов информационного обеспечения стратегического планирования,основанного на объектном подходе.

Уметь:

использовать промышленные стандартизированные решения, опирающиеся на современные технологии;

проектировать информационные системы стратегическеого планирования от этапа постановки задачи до программной реализации.

Владеть:

методами анализа информационных ресурсов; разработки различных моделей данных; конструирования программных модулей; анализа проектных решений.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическое занятие 1

Разработка диаграмм унифицированного языка моделирования

Применение UML 2.0 позволяет разделить проблему моделирования сложной системы на составные части с помощью четырех представлений:

-статическое структурное представление модели описывает структурные аспекты системы, например, с помощью диаграммы классов;

-представление взаимодействия используется для моделирования последовательностей действий и коммуникаций, описывающих кооперацию взаимодействующих экземпляров;

-представление деятельности используется для создания моделей, описывающих поток «деятельностей» в системе;

-представление в виде конечного автомата используется для описания поведения системы в терминах состояний и переходов между ними.

Эти представления не являются полностью ортогональными: концепции, используемые в одном из них, часто зависят от концепций, применяемых в другом. Так, классификаторы участников

взаимодействия должны быть определены в статической структурной модели. Такие зависимости определяются в метамодели UML, и инструментальные средства могут их задействовать для определения согласованности информации во всех представлениях системы.

Графические обозначения отдельных элементов моделей будут представлены при создании диаграмм в дальнейшем. Рассмотрим краткую характеристику диаграмм UML 2.0.

Структурные диаграммы

Диаграмма классов(Class diagram) — статическая структурная диаграмма, описывающая структуру системы, она демонстрирует классы системы, их атрибуты, методы и зависимости между классами.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

-концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;

-точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;

-точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).

Диаграмма компонентов(Component diagram) — статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п. Диаграмма компонента показывает структурные отношения между компонентами будущей информационной системы. В UML 2.0 компоненты являются автономными инкапсулированными единицами (unites) внутри системы или подсистемы, которые обеспечивают один или несколько интерфейсов. Поэтому диаграмма компонента позволяет архитектору убедиться в том, что компоненты реализуют заданную функциональность системы.

Диаграмма композитной/составной структуры (Composite structure diagram) — статическая структурная диаграмма, демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса. Подвидом диаграмм композитной структуры являются *диаграммы кооперации* (Collaboration diagram, введены в UML 2.0), которые показывают роли и взаимодействие классов в рамках кооперации. Кооперации удобны при моделировании шаблонов проектирования. Диаграммы композитной структуры могут использоваться совместно с диаграммами классов.

Диаграмма развёртывания(Deployment diagram) — служит для моделирования работающих узлов(аппаратных средств) и артефактов, развёрнутых на них. В UML 2 на узлах разворачиваются артефакты (*artifact*), в то время как в UML 1 на узлах разворачивались компоненты. Между артефактом и логическим элементом (компонентом), который он реализует, устанавливается зависимость манифестации.

Диаграмма объектов (Object diagram) — демонстрирует полный или частичный снимок моделируемой системы в заданный момент времени. На диаграмме объектов отображаются экземпляры классов (объекты) системы с указанием текущих значений их атрибутов и связей между объектами.

Диаграмма пакетов(Package diagram) — структурная диаграмма, основным содержанием которой являются пакеты и отношения между ними. Диаграммы пакетов служат, в первую очередь, для организации элементов в группы по какому-либо признаку с целью упрощения структуры и организации работы с моделью системы.

Диаграммы поведения

Диаграмма деятельности(Activity diagram) — диаграмма, на которой показано разложение некоторой *деятельности* на её составные части. Под деятельностью (англ. *activity*) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных *действий* (англ. *action*), соединённых между собой потоками, которые идут от выходов одного узла ко входам другого. Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений. Аналогом диаграмм деятельности являются схемы алгоритмов по ГОСТ 19.701-90.

Диаграмма автомата (State Machine diagram, *диаграмма конечного автомата*, *диаграмма состояний*) — диаграмма, на которой представлен *конечный автомат* с простыми состояниями,

переходами и композитными состояниями. Конечный автомат — спецификация последовательности состояний, через которые проходит объект или взаимодействие в ответ на события своей жизни, а также ответные действия объекта на эти события. Конечный автомат прикреплён к исходному элементу (классу, кооперации или методу) и служит для определения поведения его экземпляров.

Диаграмма вариантов использования(Use case diagram) —представляет собой отражение действующих лиц (актантов), которые взаимодействуют с системой, и реакцию программных объектов на их действия. Актантами могут быть как пользователи, так и внешние агенты, которым необходимо передать или получить от них информацию. Значок варианта использования отражает реакцию системы на внешнее воздействие и показывает, что должно быть сделано для актанта. Основная задача — представлять собой единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.

Диаграммы взаимодействия

Диаграмма последовательности (Sequence diagram) — диаграмма, на которой изображено упорядоченное во времени взаимодействие объектов. В частности, на ней изображаются участвующие во взаимодействии объекты и последовательность сообщений, которыми они обмениваются. Диаграмма последовательности показывает хронологическую последовательность сообщений между объектами во взаимодействии. Она состоит из нескольких участников, таких как агенты, системы или подсистемы, классы и компоненты, представленные линиями жизни (lifelines), а также сообщения, которыми они обмениваются при взаимодействии.

Диаграмма коммуникации (Communication diagram, в UML 1.x — *диаграмма кооперации*, *collaboration diagram*) — диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. В отличие от диаграммы последовательности, на диаграмме коммуникации явно указываются отношения между элементами (объектами), а время как отдельное измерение не используется (применяются порядковые номера вызовов). Диаграмма коммуникации показывает поток сообщений между объектами и то, как несколько объектов сотрудничают при выполнении общей задачи. Как и диаграмма последовательности (sequence diagram), диаграмма коммуникации тоже может моделировать динамическое поведение для варианта использования (use case). Однако диаграмма коммуникации больше нацелена на показ того, как происходит координация, чем на хронометрирование последовательности.

Диаграммы коммуникации и последовательности транзитивны, выражают взаимодействие, но показывают его различными способами и с достаточной степенью точности могут быть преобразованы одна в другую.

Примечание.

По причине того, что диаграммы коммуникации и последовательности являются разными взглядами на одни и те же процессы, Rational Rose позволяет создавать из диаграммы коммуникации диаграмму последовательности и наоборот, а также производит автоматическую синхронизацию этих диаграмм.

Диаграмма обзора взаимодействия (Interaction overview diagram) — разновидность диаграммы деятельности, включающая фрагменты диаграммы последовательности и конструкции потока управления. Этот тип диаграмм включает в себя диаграммы последовательностей действий и диаграммы сотрудничества. Эти диаграммы позволяют с разных точек зрения рассмотреть взаимодействие объектов в создаваемой системе.

Диаграмма синхронизации(Timing diagram) — альтернативное представление диаграммы последовательности, явным образом показывающее изменения состояния на линии жизни с заданной шкалой времени. Эта диаграмма может быть полезна в приложениях реального времени.

Таким образом, выбранный разработчиком набор диаграмм позволяет создать практически любое представление о проектируемой системе.

Примечание.

Изображая диаграмму, воспользуйтесь следующими рекомендациями:

-дайте диаграмме имя, соответствующее ее назначению;

-расположите элементы так, чтобы свести к минимуму число пересечений;

-пространственно элементы расположите так, чтобы семантически близкие сущности располагались на диаграмме рядом;

-используйте примечания и цвет, чтобы привлечь внимание читателя к важным особенностям диаграммы.

Унифицированный процесс разработки программного обеспечения

Процесс проектирования ИСТ, кроме основных концепций и понятий, используемых при проектировании и реализации ИСТ, включает в себя технологию проектирования. Одной из развитых современных технологий является унифицированный процесс(Rational Unified Process,RUP). RUP – одна из лучших технологий разработки программного обеспечения, созданная в компании Rational Software, входящей в состав IBM. Унифицированный процесс позволяет создавать сложные программные системы, основываясь на индустриальных методах разработки [2, 3].

Вся разработка информационной системы (ИС) рассматривается в RUP как процесс создания артефактов. Любой результат работы проекта, будь то исходные тексты, объектные модули, документы, передаваемые пользователю, модели – это подклассы всех артефактов проекта.

Одним из интереснейших классов артефактов проекта являются модели, которые позволяют разработчикам определять, визуализировать, конструировать и документировать артефакты программных систем.

Модели позволяют рассмотреть будущую систему, ее объекты и их взаимодействие еще до вкладывания значительных средств в разработку, позволяют увидеть ее глазами будущих пользователей снаружи и разработчиков изнутри еще до создания первой строки исходного кода. Большинство моделей представляются диаграммами на унифицированном языке моделирования UML.

Основными моделями, создаваемыми в RUP, являются: модель вариантов использования, модель анализа, модель проектирования и модель реализации. Эти модели являются результатом основных работ процесса, к которым относятся: определение требований, анализ, проектирование и реализация. Рассмотрим содержание каждого из них.

1.3.1 Определение требований

Одним из важнейших этапов разработки ИС, согласно RUP, является этап определения требований, который заключается в сборе всех возможных пожеланий заказчика к работе системы. На данном этапе в ходе интервью с пользователями и изучения документов, аналитики должны собрать как можно больше требований к будущей системе, что не так просто, как кажется на первый взгляд. Позднее эти данные должны будут систематизированы и структурированы. Для того чтобы верно определить требования, разработчики должны понимать **контекст** (часть предметной области) в котором будет работать будущая система.

Определение контекста информационной системы

Для определения контекста ИСТ выполняется предпроектное обследование предметной области(области использования ИСТ). Для этого создаются модель предметной области и бизнес-модель, что является различными подходами к одному и тому же вопросу. Часто создается что-то одно: модель предметной области или бизнес-модель[2].

Отличия этих моделей в том, что модель предметной области описывает важные понятия, с которыми будет работать система и связи их между собой. Тогда как бизнес-модель описывает бизнес-процессы (существующие или будущие), которые должна автоматизировать(поддерживать) система. Поэтому кроме определения бизнес-объектов, вовлеченных в процесс, эта модель определяет работников, их обязанности и действия, которые они должны выполнять.

Использование UML не ограничивается моделированием программного обеспечения. Его также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Для создания модели предметной области с помощью UML используется обычная диаграмма классов, однако для создания бизнес-модели ее уже явно недостаточно. В этом случае применяется диаграмма вариантов использования с использованием дополнительных значков, которые отражают сущность бизнес-процессов – это бизнес-актант, бизнес-прецедент, бизнес-сущность и бизнес-управление. Эта модель намного ближе к следующей модели, создаваемой в процессе разработки – модели анализа.

При моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений часто используются диаграммы деятельности.

На практике диаграммы деятельности применяются в основном двумя способами:

-для моделирования процессов;

-для моделирования операций.

В первом случае внимание фокусируется на деятельности с точки зрения действующих лиц, которые работают с системой. Важным здесь является применимость диаграмм деятельности для описания бизнес-процессов. В данном случае для построения диаграмм деятельности используется так называемая траектория объекта, или поток объекта(object flow). Суть его состоит в том, что на диаграмме кроме деятельности можно изобразить и объекты, относящиеся к деятельности. С помощью символа зависимости(пунктирная стрелка) эти объекты можно соотнести с той деятельностью или переходом, где они создаются, изменяются или уничтожаются. Траектория объекта позволяет показать объекты, относящиеся к деятельности, а также моменты переходов этих объектов из одного состояния в другое.

Рекомендации по построению диаграмм деятельности для моделирования процессов заключаются в следующем.

Моделируют бизнес-процессы в несколько этапов, первым из которых является разбиение их на подпроцессы. Подпроцессы, являющиеся "участками большого процесса", описать легче.

Дальше выделяют ключевые объекты (и создают для них дорожки), определяют предусловия и постусловия каждого процесса (т. е. его границы), описывают деятельности и переходы, отображают на диаграммах состояния ключевых объектов, в которые они переходят в ходе процесса.

В итоге создается не какая-то абстрактная диаграмма, а модель реального бизнес-процесса в реальной компании, занимающейся реальным бизнесом.

Пример детализации конкретного бизнес-процесса с помощью диаграммы деятельности, созданной в Rational Rose, показан на рисунке 1. На диаграмме отражена деятельность выдачи товара со склада [1].

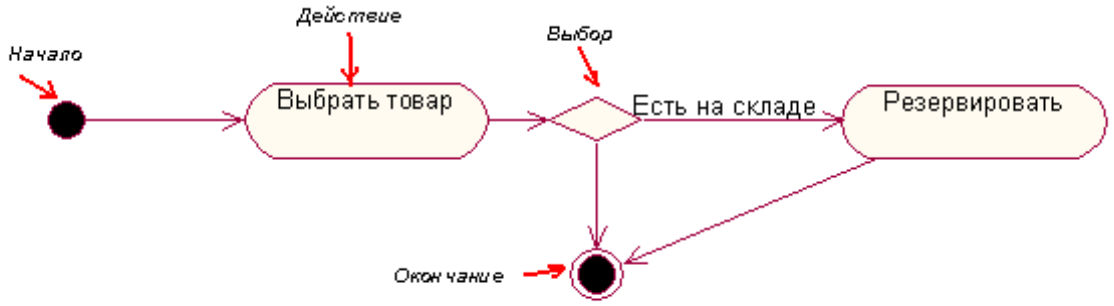


Рисунок 1 - Пример диаграммы активности, созданной в Rational Rose

Второй пример моделирования бизнес процесса оформление заказа в Интернет-магазине представлен на рисунке 2.

Рисунок 2 – Оформление заказа в Интернет-магазине

На рисунке 3 представлена диаграмма деятельности, выполненная в Borland Together. Здесь показана параметризованная деятельность с объектным узлом параметра, соединенным с контактами действий. Объектные узлы параметров размещаются на границе диаграммы, и ребра потока объектов соединяют их с контактами. Тип объекта, удерживаемого в объектном узле, обычно отображается в метке этого узла. В данном примере информация, используемая для заполнения заказа, предоставляется как входной параметр и передается в действие по вызову работы “Заполнение заказа”.

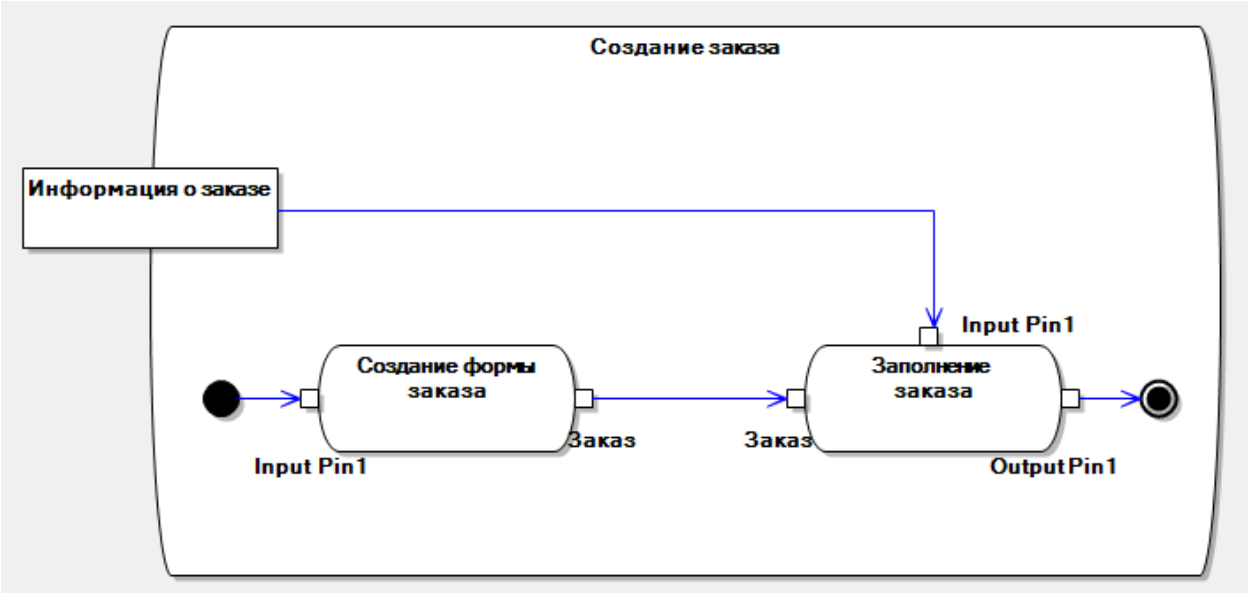


Рисунок 3 – Диаграмма активности, выполненная в Borland Together

Узлы управления в начале и в конце потока на рисунке 2 — это, соответственно, исходный и заключительный узлы. Когда вызывается деятельность “Создание заказа”, управляющий маркер помещается в начальный узел, а маркер данных с информацией о заказе — в объектный узел входного параметра. Управляющий маркер двигается от исходного узла к действию “Создание формы заказа”, которое начинает выполняться. Маркер данных передается от соответствующего параметра действию, вызывающему “Заполнение заказа”, которому приходится ждать до начала выполнения, пока “Создание формы заказа” не предоставит ему другие входные данные. После завершения действия “Заполнение заказа” управляющий маркер передается на конечный узел, деятельность завершается, а управление возвращается элементу, инициировавшему эту деятельность.

В случае моделирования операций диаграммы деятельности играют роль "продвинутых" блок-схем и применяются для подробного моделирования вычислений. На первое место при таком использовании выходят конструкции принятия решения, а также разделения и слияния потоков управления (синхронизации). Этот способ применяется при детализации вариантов использования и других процедур.

Рекомендации по построению диаграмм деятельности для моделирования операций заключаются в следующем.

Процесс построения диаграммы деятельности можно описать в виде последовательности таких действий:

1)Составление перечня деятельностей в системе

Как исходные данные для этой операции хорошо подходит список вариантов использования(или список операций). Дополняться диаграммой деятельности может каждый сценарий использования. Можно также попытаться описать связь между ними.

2)Определение зависимостей между деятельностями

Для каждой деятельности нужно найти деятельности, непосредственно предшествующие (и следующие за ней тоже), то есть деятельности, без выполнения которых поток управления не может перейти к данной деятельности.

3)Выделение параллельных потоков деятельностей

Выделяются деятельности, имеющие общих предшественников.

4)Определение условий переходов

Для этого формулируются выражения, которые могут принимать только два значения - "истинно" или "ложно", соответствующие альтернативным потокам управления.

5)Уточнение сложных деятельностей

Повторяя пункты 1-4 для каждой из деятельностей(при необходимости), можно уточнить сложные деятельности.

Таким образом, диаграммы деятельности в UML используются для моделирования потоков различного типа: потоков сигналов или данных, а также алгоритмических или процедурных потоков.

Пример диаграммы деятельности, выполненной в MS Office Visio, представлен на рисунке 4.

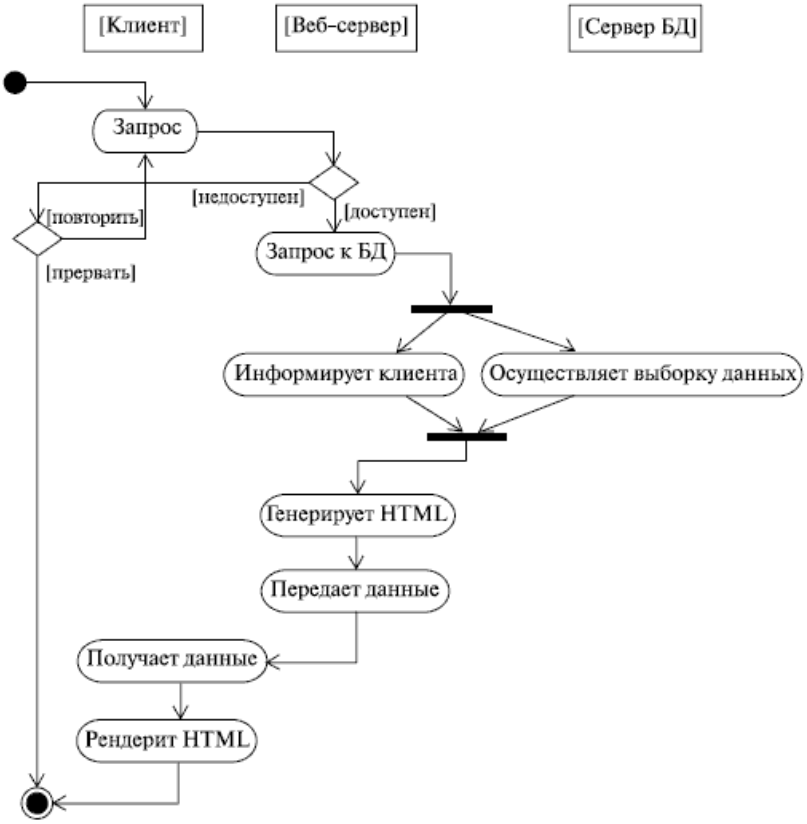


Рисунок 4 - Диаграмма деятельности, выполненная в MS Office Visio

На диаграмме показана работа с веб-приложением, решающим некую задачу в удаленной базе данных. Привлекает внимание расположение деятельностей на этой диаграмме: они как бы разбросаны по трем дорожкам, каждая из которых соответствует поведению одного из трех объектов - клиента, веб-

сервера и сервера баз данных. Благодаря этому легко определить, каким из объектов выполняется каждая из активностей, что очень упрощает ее восприятие.

Аналогия с дорожками действительно очень удачна. Именно таково официальное название элемента нотации UML, позволяющего указать распределение ролей на диаграмме деятельности.

Создавая диаграммы деятельности, необходимо учитывать, что они лишь моделируют срез некоторых динамических аспектов поведения системы. С помощью единственной диаграммы деятельности никогда не удастся охватить все динамические аспекты системы. Вместо этого следует использовать разные диаграммы деятельности для моделирования динамики рабочих процессов или отдельных операций.

Спецификация требований к информационной системе

Основным средством спецификации требований к проектируемой информационной системе в рамках RUP является модель вариантов использования. Главное назначение диаграммы вариантов использования заключается в формализации функциональных требований к системе. Основная задача — представить единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.

Модель варианта использования дает подробную информацию о поведении системы или приложения, которое разрабатывается. Она определяет требования к системе в терминах требующейся функциональности (вариантов использования) для достижения целей или для решения проблемы, определенной пользователем. Она же описывает окружение(агенты) и отношения между вариантами использования и агентами. Модель вариантов использования обычно включает в себя диаграммы вариантов использования и диаграммы действий, которые описывают то, как пользователи общаются с системой.

Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия внутренней структуры этой сущности. В качестве такой сущности может выступать исходная система или любой другой элемент модели, который обладает собственным поведением, подобно подсистеме или классу в модели системы.

Каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемую сущность или систему по запросу пользователя (актера), т. е. определяет способ применения этой сущности. Сервис, который инициализируется по запросу пользователя, представляет собой законченную последовательность действий. Это означает, что после того как система закончит обработку запроса пользователя, она должна возвратиться в исходное состояние, в котором готова к выполнению следующих запросов.

Примерами вариантов использования могут являться следующие действия: проверка состояния текущего счета клиента, оформление заказа на покупку товара, получение дополнительной информации о кредитоспособности клиента, отображение графической формы на экране монитора и другие действия.

Пример простейшей диаграммы вариантов использования “Заказ товара”, выполненной в Rational Rose, представлен на рисунке 5. На диаграмме показаны условные графические изображения главных элементов диаграммы – действующего лица(актанта) и варианта использования, а также связи между ними.

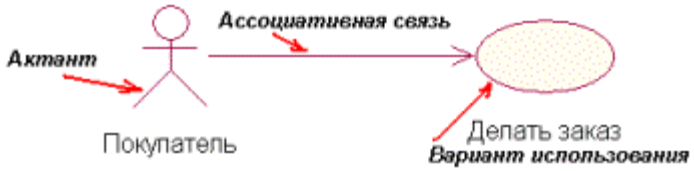


Рисунок 5 - Диаграмма вариантов использования,
выполненная в Rational Rose

Пример диаграммы вариантов использования, выполненной в Borland Together, показан на рисунке 6.

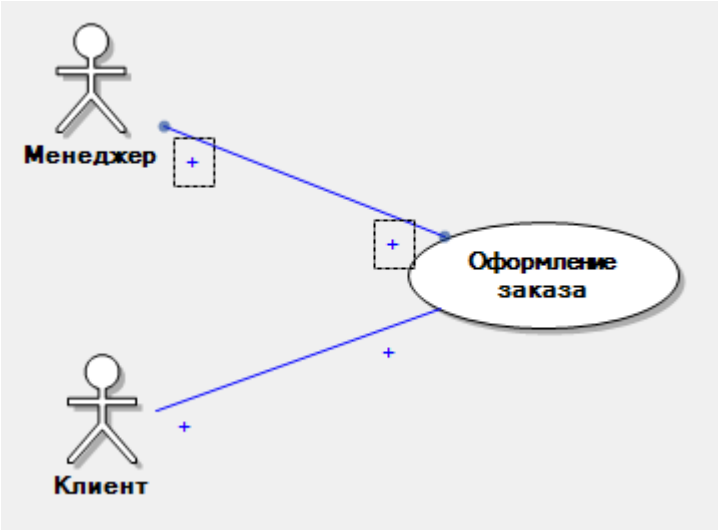


Рисунок 6 – Диаграмма вариантов использования, выполненная в Borland Together

Далее после создания диаграммы вариантов использования следует определить реализацию каждого варианта использования. Для этого применяются следующие способы:

-текстовое описание;

-описание алгоритма с помощью диаграмм деятельности;

-создание одной или несколько диаграмм взаимодействия.

Текстовое описание, соответствующее основной модели в рамках RUP, представляет собой:

-краткое описание;

-действующие лица;

-специальные требования;

-предпосылки;

-постусловия;

-точки расширения.

Например, рассмотрим вариант использования “Сделать предложение на аукционе”, диаграмма которого представлена на рисунке 7.

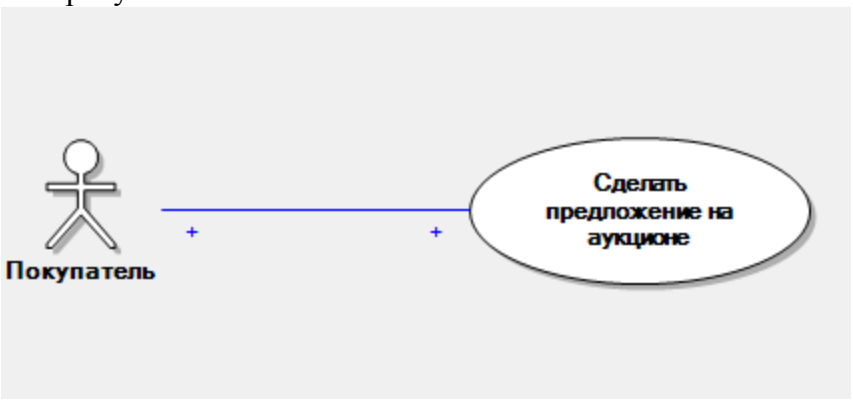


Рисунок 7 – Пример диаграммы варианта использования

В случае с приложением по ведению аукциона, речь может идти о представленном ниже потоке событий(последовательности, инициированной действующим лицом при подаче заявки в системе аукциона):

Основной поток:

-заявка (предложение цены): прецедент начинается в тот момент, когда покупатель предлагает свою цену на текущую позицию;

-ввод суммы: покупатель вводит сумму предложения. Система подтверждает, что сумма предложения превышает текущую ставку на значение, кратное шагу заявки для данной позиции;

-покупатель подтверждает заявку: покупатель подтверждает свое намерение разместить заявку;

-обработка заявки: система добавляет заявку к данной позиции;

-подтверждение заявки: система подтверждает наличие заявки путем отправки покупателю электронного сообщения. Продавец также уведомляется по электронной почте.

Альтернативные потоки операций могут описывать, что произойдет, если прием заявок будет прекращен до подачи предложения, если сумма заявки будет признана недействительной или покупатель не подтвердит подачу заявки.

Кроме текстового описания для детализации конкретного варианта использования(прецедента) можно построить диаграмму деятельности, правила построения которых аналогичны рассмотренных ранее.

Один из основных способов представления реализации варианта использования является создать одну или несколько диаграмм взаимодействия в форме диаграмм коммуникации или диаграмм последовательности, которые описывают один или несколько сценариев данного варианта использования. Этот способ в наибольшей степени соответствует идеологии UML и рекомендуется как основной и предпочтительный. Все эти операции выполняются на этапе анализа.

1.3.2 Анализ

После определения контекста, в котором будет работать система и требований к ней, наступает черед анализа полученных данных. В процессе анализа создается **аналитическая модель(модель анализа),** которая подводит разработчиков к архитектуре будущей системы. Аналитическая модель – это взгляд на систему изнутри, в отличие от модели вариантов использования, которая показывает, как система будет выглядеть снаружи.

Эта модель позволяет понять, как система должна быть спроектирована, какие в ней должны быть классы и как они должны взаимодействовать между собой. Основное ее назначение - определить направление реализации функциональности, выявленной на этапе сбора требований и сделать набросок архитектуры системы.

Модель анализа описывает логическую структуру системы и является фундаментом модели проектирования. Но в отличие от создаваемой в дальнейшем модели проектирования, модель анализа является в большей степени концептуальной моделью и только приближает разработчиков к классам реализации. Эта модель не должна иметь возможных противоречий, которые могут встретиться в модели вариантов использования.

Для построения аналитической модели выполняется анализ вариантов использования, который включает в себя:

-идентификацию классов, участвующих в реализации потоков событий;

-определение обязанностей классов;

-определение атрибутов и ассоциаций классов;

-унификацию классов анализа.

В потоках событий варианта использования выявляются классы трех типов:

-граничные классы, являющиеся посредниками при взаимодействии с внешними объектами;

-классы-сущности, представляющие собой основные абстракции (понятия) разрабатываемой системы;

-управляющие классы, обеспечивающие координацию поведения объектов в системе.

Классы анализа отражают функциональные требования к системе и моделируют объекты предметной области. Совокупность классов анализа представляет собой начальную концептуальную модель системы.

Для отображения модели анализа при помощи UML используется диаграмма классов со стереотипами (образцами поведения) «граничный класс», «сущность», «управление», а для детализации используются диаграммы взаимодействия, которые описывают взаимодействие групп объектов в различных условиях их поведения. Наиболее используемым типом таких диаграмм являются диаграммы коммуникации и последовательности.

Диаграмма коммуникации

Диаграмма коммуникации делает фокус на представлении группы взаимодействующих объектов и связей между ними, образующихся, если объекты общаются друг с другом посредством отсылки и приема сообщений. Также диаграммы коммуникаций подобны диаграммам объектов, но на них дополнительно могут быть показаны отсылаемые сообщения, причем допускается даже с указанием нумерации, описывающей порядок их следования во времени.

Диаграмма коммуникации используется для описания поведения системы как последовательности обмена сообщениями между элементами.

Основные сущности, используемые на диаграмме:

-роли, которые играют взаимодействующие элементы;

-объекты – экземпляры конкретных классов;

- связи - отношения, соединяющие взаимодействующие элементы.

Диаграмма коммуникации описывает поведение как взаимодействие, т. е. как протокол обмена сообщений между объектами.

Построение диаграммы коммуникации (кооперации) можно начинать сразу после построения диаграммы вариантов использования. В этом случае каждый из вариантов использования может быть специфицирован в виде отдельной диаграммы кооперации уровня спецификации.

Главная особенность диаграммы коммуникации (кооперации, сотрудничества) заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Прежде всего, на диаграмме коммуникации(кооперации, сотрудничества) в виде прямоугольников(окружностей) изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее должны быть изображены динамические связи - потоки сообщений. Они представляются в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

Пример диаграммы коммуникации (сотрудничества) показан на рисунке 8. Здесь показаны объект класса сущности “счет”, объект класса управления “обработчик” и объект граничного класса “интерфейс запроса на оплату”.

Стереотип «граничный класс» отображает класс, который взаимодействует с внешними актантами, «сущность» – отображает классы, которые являются хранилищами данных, а «управление» – классы, управляющие запросами к сущностям.

Линии, соединяющие объекты классов, отражают их взаимодействие.

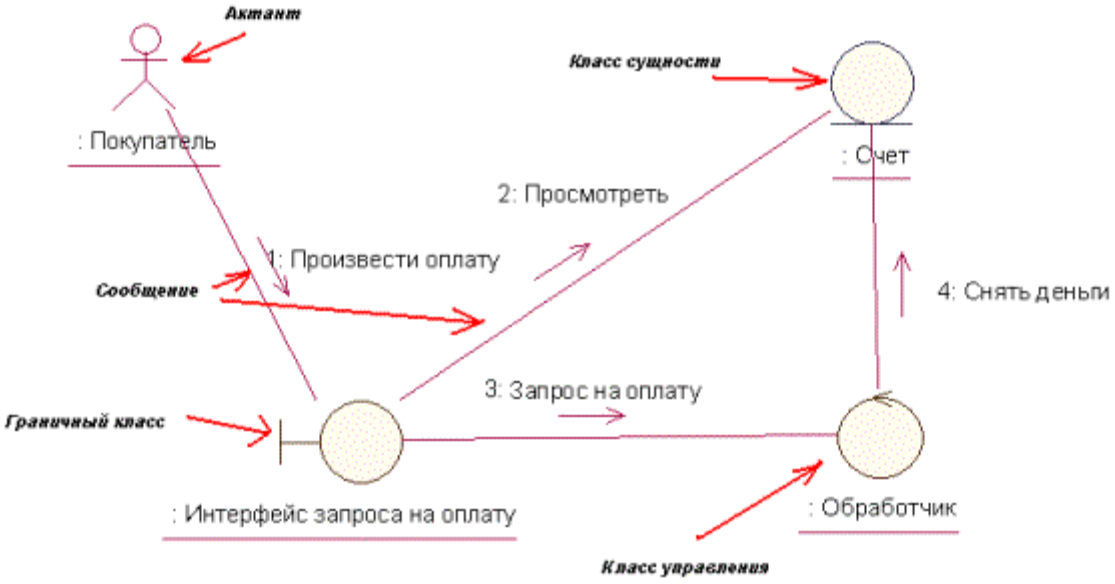


Рисунок 8 - Пример диаграммы коммуникации (сотрудничества)

Нумерация сообщений показывает их порядок, однако назначение диаграммы не в том, чтобы рассмотреть порядок обмена сообщениями, а в том, чтобы наглядно показать связи классов друг с другом.

При создании диаграммы коммуникации можно явно указать имена ассоциаций и ролей, которые играют объекты в данной ассоциации, как показано на диаграмме, изображенной на рисунке 9. Здесь показаны ассоциации “Продажа товара” и “Продажа компьютера”, а также роли “ клиент” и “менеджер”.

Рисунок 9 - Пример изображения ассоциаций и ролей

Диаграмма последовательности

Если акцентировать внимание на порядке взаимодействия, то другим его представлением будет диаграмма последовательности (Sequence).

Диаграмма последовательности - это диаграмма, чаще всего, описывающая один сценарий приложения. На диаграмме изображаются экземпляры объектов и сообщения, которыми они обмениваются в рамках одного варианта использования. Участники диаграммы именуются следующим образом: **имя: Класс**, где и имя, и класс являются не обязательными, но если используется класс, то присутствие двоеточия обязательно.

На диаграмме последовательности, каждый участник представлен вместе со своей линией жизни (lifeline), это вертикальная линия под объектом, вертикально упорядочивающая сообщения на странице. Важно: все сообщения на диаграмме следует читать сверху вниз. Каждая линия жизни имеет полосу активности (прямоугольники), которая показывает интервал активности каждого участника при взаимодействии.

Обозначение различных сообщений на диаграмме показано на рисунке 10.

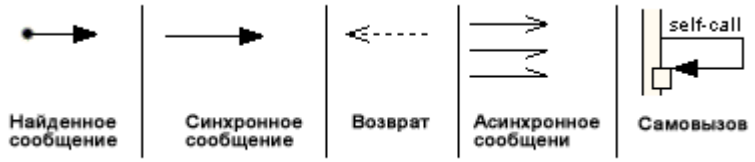


Рисунок 10 - Обозначение различных сообщений на диаграмме

У первого сообщения нет участника, пославшего его, поскольку оно приходит от неизвестного источника. Такое сообщение называется найденным сообщением (found message). Отправитель или получатель сообщения может находиться за пределами диаграммы коммуникации, и в этом случае используют входной и выходной шлюзы.

Сообщения, которыми обмениваются элементы, могут быть синхронными или асинхронными, что отражается в нотации стрелочек. Синхронные (synchronous message) - требующие возврата ответа, а асинхронные (asynchronous message) – ответа не требуют(вызывающий объект может продолжать работу). На диаграмме синхронные вызовы обозначаются закрашенными стрелочками, асинхронные – не закрашенными или половинными стрелочками.

Обратной пунктирной стрелкой показывается возврат ответа на сообщение(если сообщение является синхронным). Лучше применять изображение возврата только в тех случаях, когда это поможет лучше понять устройство взаимодействия. Во всех остальных случаях, стоит опускать изображения возвратов, т.к. они будут вносить некоторую неразбериху. Просто, при использовании синхронного сообщения, стоит помнить, что у него всегда есть возврат.

Если элемент диаграммы связан сам с собой, то такая связь называется рефлексивной(самовызов).

Если в сообщении требуется передать параметры, то они указываются в скобках через запятую, с указанием типа параметра (messageText(text : string)).

Для того чтобы задать порядок следования сообщений, используют десятичную нумерацию.

Пример диаграммы последовательности показан на рисунке 11. На ней представлены два объекта и все возможные виды сообщений.

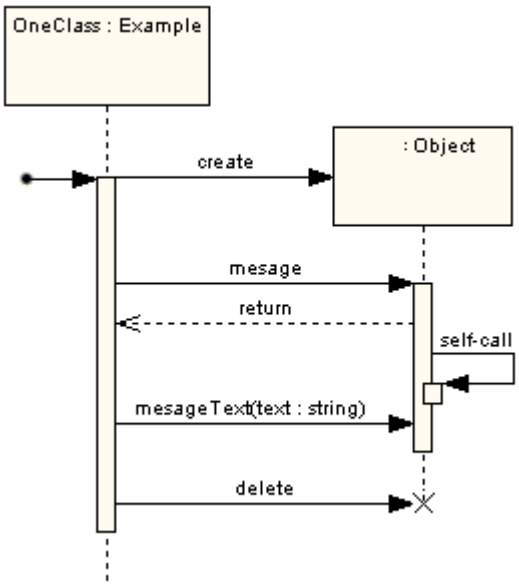


Рисунок 11 - Виды сообщений, которыми обмениваются объекты

Диаграмма последовательности, соответствующая диаграмме сотрудничества, показанной на рисунке 8, представлена на рисунке 12. Виды сообщений, которыми обмениваются объекты, аналогичны сообщениям диаграммы коммуникации, изображенным на рисунке 10.

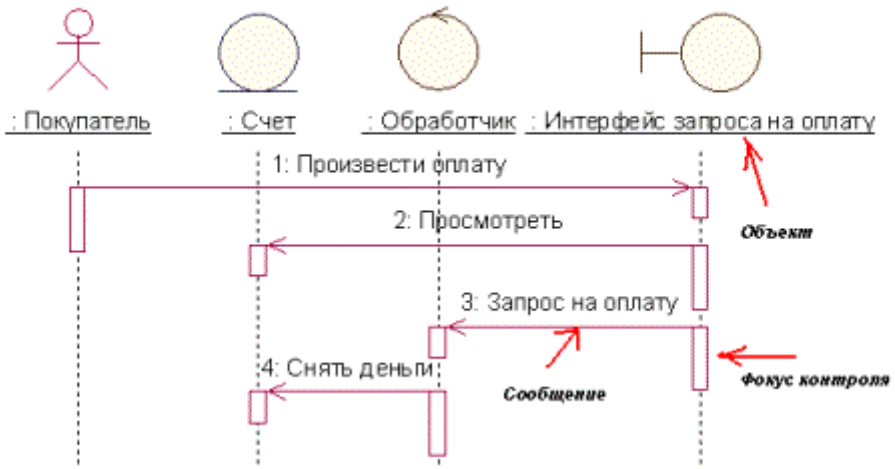


Рисунок 12 - Пример диаграммы последовательности действий

Решение о том какую из двух диаграмм нужно создавать первой, зависит от предпочтений конкретного разработчика. Поскольку эти диаграммы являются отображением одного и того же процесса, то и та и другая позволяют отразить взаимодействие между объектами.

Диаграммы коммуникации и последовательности транзитивны, выражают взаимодействие, но показывают его различными способами и с достаточной степенью точности могут быть преобразованы одна в другую.

При использовании такого инструмента для создания моделей как Rational Rose, эти два вида диаграмм могут быть созданы друг из друга автоматически [5].

В некоторых случаях могут строиться диаграммы обзора взаимодействия и диаграммы синхронизации.

1.3.3 Проектирование

Следующим этапом в процессе создания системы будет проектирование, в ходе которого на основании моделей, созданных ранее, создается **модель проектирования**. Эта модель отражает физическую реализации системы и описывает создаваемый продукт на уровне классов и компонентов. В отличие от модели анализа, модель проектирования имеет явно выраженную зависимость от условий реализации, применяемых языков программирования и компонентов.

Модель проектирования, используя различные диаграммы (в том числе диаграммы последовательности, машины состояний, компонента и размещения), подробно описывает, как устроено приложение и как оно будет реализовываться. Она также описывает структурные компоненты программ и технологий, например, обеспечивающих персистентность, распределение, безопасность и доступ к данным.

Для максимально точного понимания архитектуры системы, эта модель должна быть максимально формализована, и поддерживаться в актуальном состоянии на протяжении всего жизненного цикла разработки системы.

В RUP проектирование концентрируется вокруг определения архитектуры системы, а для систем с большой долей программного обеспечения - вокруг архитектуры программного обеспечения. Использование компонентных архитектур - один из шести наилучших подходов к разработке программ, инкорпорированных в RUP, рекомендует уделять больше времени на разработку и сопровождение архитектур. Время, затраченное на эти усилия, сокращает риски, связанные с ненадежными и негибкими системами.

Для создания модели проектирования используются целый набор UML диаграмм: диаграммы классов, диаграммы композитной структуры(кооперации), диаграммы взаимодействия, диаграммы активности. Основной является диаграмма классов.

Диаграмма классов

Диаграмма классов является основным типом диаграммы статической структуры. Она описывает структуру системы, показывая её классы, их атрибуты и операторы, и также взаимосвязи этих классов.

Каждый класс имеет имя, размещенное в верхнем блоке прямоугольника, изображающего класс. Для атрибутов и операций в элементах отводится отдельный блок. Каждый блок разделяется горизонтальной чертой.

Для атрибутов и операций применяются спецификаторы доступа. Спецификатора доступа языка C++ (public, private, protected) в UML отображаются символами + (public), - (private), # (protected), которые ставятся перед именем атрибута/операции. Также возможен вариант с ключевыми словами public, private, protected. Значение спецификаторов доступа: public - поля/методы класса видны снаружи класса. Т.е. к ним могут получать доступ объекты класса. private - поля/методы класса видны только внутри определения класса. protected - поля/методы класса видны в определении самого класса и в определениях производных классов.

Между классами существуют различные виды взаимодействия (или связи): один класс может быть производным другого, третий может содержать объект четвёртого в виде поля и т.д. Для различных видов взаимодействия в UML есть специальные названия.

Первый вид взаимодействия - ассоциация(association).

Ассоциация – это семейство связей двух и более классов. Обычно ассоциация возникает, когда один класс вызывает метод другого или если при вызове метода в качестве аргумента передаётся объект другого класса. Иногда при ассоциации показывают направленность (если это имеет значение).

Частным случаем ассоциации является связь – простая взаимосвязь между объектами. Она представляется линией соединяющей два или более объектных блока. Она встречается на диаграммах классов или объектов.

Всего существует пять типов ассоциации. Но наиболее распространены два: двунаправленная и однонаправленная ассоциации.

Сообщение направленная ассоциация(Message/Directed Association) используется, когда один класс “общается” с другим при создании экземпляра класса. Экземпляр класса — это описание конкретного объекта в памяти. Класс описывает свойства и методы, которые будет доступны у объекта, построенного по описанию, заложенному в класс. Экземпляры используют для представления конкретных сущностей реального мира.

Графически направленная ассоциация представляется в виде стрелочки направленной к “вызываемому” классу.

Частными вариантами ассоциации являются: агрегация и композиция.

Агрегация(быть частью) применяется, когда один класс должен быть контейнером других классов. Причем время существования содержащихся классов никак не зависит от времени существования класса контейнера. Графически агрегация представляется пустым ромбиком на блоке класса и линией, идущей от этого ромбика к содержащемуся классу.

Композиция - еще один случай ассоциации, но более строгий. В отличии от агрегации, композиция имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер будет уничтожен, то всё его содержимое будет уничтожено также. Графически представляется, как и агрегация, но с закрашенным ромбиком.

Различие между этими двумя видами ассоциации состоит в том, что композиция может быть частью одного и только одного целого, в то время как агрегация может быть частью нескольких объектов.

Еще одной разновидностью связи является **генерализация** (обобщение). Генерализация показывает, что один из двух связанных классов (*подтип*), является более частной формой другого (*супертип*), который называется обобщением первого. Графически генерализация представляется линией с пустым треугольником у супертипа.

Последнее отношение, которое мы рассмотрим, будет **реализация**(realization). Данная связь показывает отношение: класс - объект. На диаграмме реализация показывается пунктирной линией и не закрашенной стрелочкой.

Одной из важнейших характеристик взаимодействия является кратность(multiplicity) роли ассоциации. Кратностью роли ассоциации называется характеристика, указывающая, сколько объектов класса с данной ролью может или должно участвовать в каждом экземпляре ассоциации.

Наиболее распространенным способом задания кратности роли ассоциации является указание конкретного числа или диапазона. Например, указание “1” говорит о том, что все объекты класса с данной ролью должны участвовать в некотором экземпляре данной ассоциации, причем в каждом экземпляре ассоциации может участвовать ровно один объект класса с данной ролью. Указание диапазона “0..1” говорит о том, что не все объекты класса с данной ролью обязаны участвовать в каком-либо экземпляре данной ассоциации, но в каждом экземпляре ассоциации может участвовать только один объект. Аналогично, указание диапазона “1..*” говорит, что все объекты класса с данной ролью должны участвовать в некотором экземпляре данной ассоциации, и в каждом экземпляре ассоциации должен участвовать хотя бы один объект (верхняя граница не задана).

Пример диаграммы классов, на которой показаны все возможные варианты связей, представлен на рисунке 13.

Проектирование классов на данном этапе включает следующие действия:

-детализация проектных классов;

-уточнение операций и атрибутов;

-уточнение связей между классами;

-моделирование состояний для классов.

Детализация проектных классов, определенных в процессе анализа, уточнение атрибутов классов заключается в следующем:

-задается тип атрибута и значение по умолчанию (необязательно);

-задается видимость атрибутов: public, private или protected;

-при необходимости определяются производные (вычисляемые) атрибуты.

Обязанности классов, определенные в процессе анализа и документированные в виде «операций анализа», преобразуются в операции, которые будут реализованы в коде. При этом:

-каждой операции присваивается краткое имя, характеризующее ее результат;

-определяется полная сигнатура операции;

-создается краткое описание операции, включая смысл всех ее параметров;

-определяется видимость операции: public, private или protected;

-определяется область действия операции: операция объекта или операция класса.

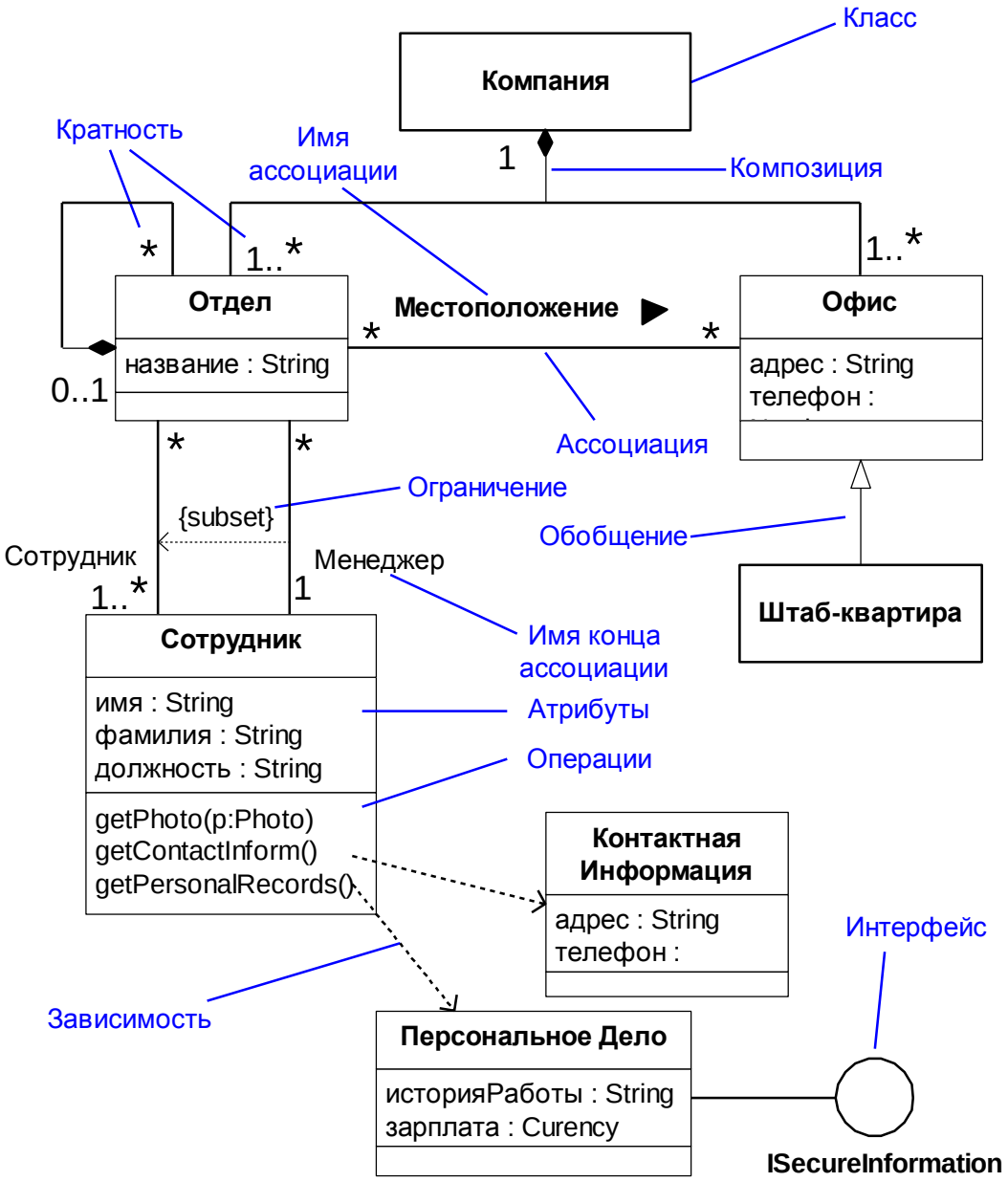


Рисунок 13 - Пример диаграммы классов

В процессе проектирования связи между классами подлежат уточнению. Ассоциации между граничными и управляющими классами преобразуются в зависимости. Агрегации, обладающие свойствами композиции, преобразуются в связи композиции. Связи обобщения могут преобразовываться в ситуациях с так называемой метаморфозой подтипов, когда объект суперкласса может менять свой подтип

Например, для бизнес-процесса – продажа товаров по заказу(каталогу), можно выделить классы Заказ и Клиент, который может представляться как юридическим, так и физическим лицом. Каждый класс имеет определенные атрибуты(свойства). Заказ выполняется в определенный день. Клиент имеет имя и т.д.

Пример диаграммы классов, созданной в среде Rational Rose, показан на рисунке 14.

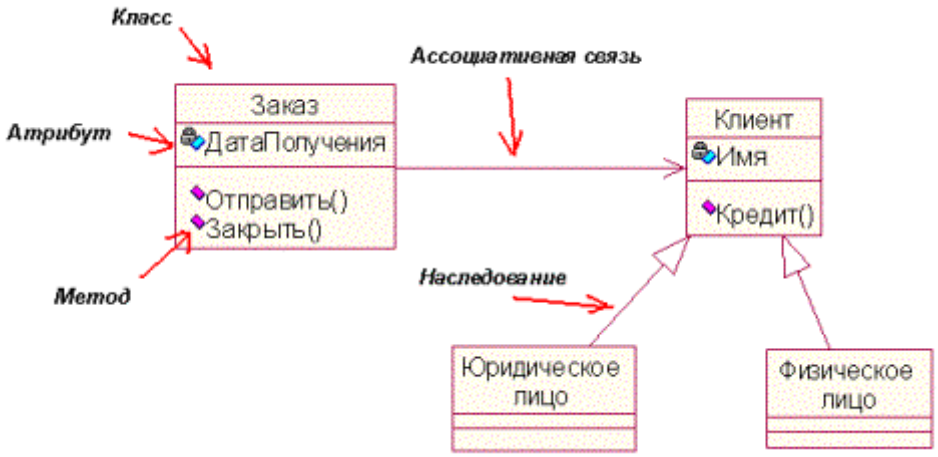


Рисунок 14 - Пример диаграммы классов, созданной в среде Rational Rose

Из диаграммы видно, что между классами “Заказ” и “Клиент” имеет место связь однонаправленная ассоциация.

Из диаграммы видно, что базовый класс “Клиент” имеет два производных класса “Юридическое лицо” и “Физическое лицо”, соединенных с базовым классом связью типа “Обобщение”, реализующей принцип наследования.

Другой вариант диаграммы классов для того же бизнес-процесса, созданной в среде Borland Together, показан на рисунке 15. На этой диаграмме классы имеют более полный набор атрибутов и операций. Однако класса Клиент не является базовым. Наследования классов на данной диаграмме не предусмотрено.

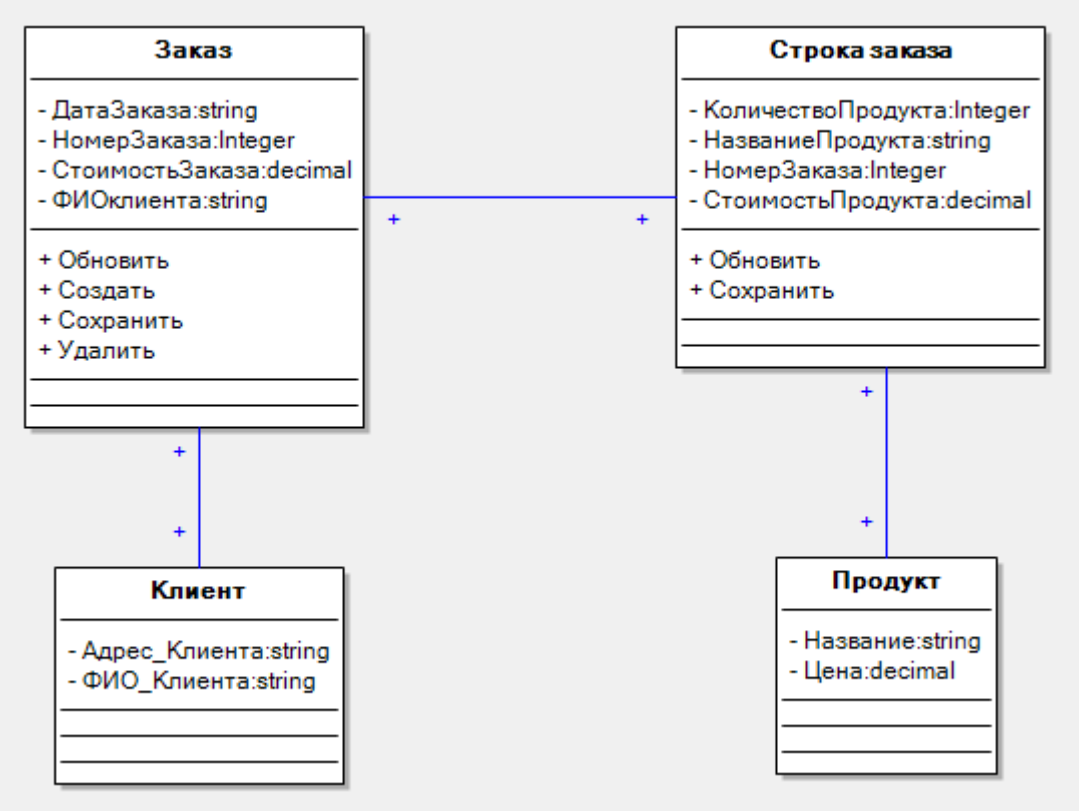


Рисунок 15 - Диаграмма классов, созданная в среде Borland Together

Основные правила построения диаграмм классов

В UML необязательно расписывать все детали классов. Это будет сделано при написании кода на конкретном языке (в нашем случае - C++). В UML-диаграмме можно опускать ненужные детали. Например, в диаграмму элемента можно добавить только те операции/атрибуты, которые важны для данной диаграммы, неважные особенности класса в UML можно опускать.

Для более полного раскрытия архитектуры проектируемой системы могут строиться диаграммы композитной/составной структуры и диаграммы автомата.

Диаграмма композитной/составной структуры

Диаграмма композитной/составной структуры — статическая структурная диаграмма, демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса.

Подвидом диаграмм композитной структуры являются диаграммы кооперации (Collaboration diagram, введены в UML 2.0), которые показывают роли и взаимодействие классов в рамках кооперации. Кооперации удобны при моделировании шаблонов проектирования.

Диаграммы композитной структуры могут использоваться совместно с диаграммами классов, как показано на рисунке 16.

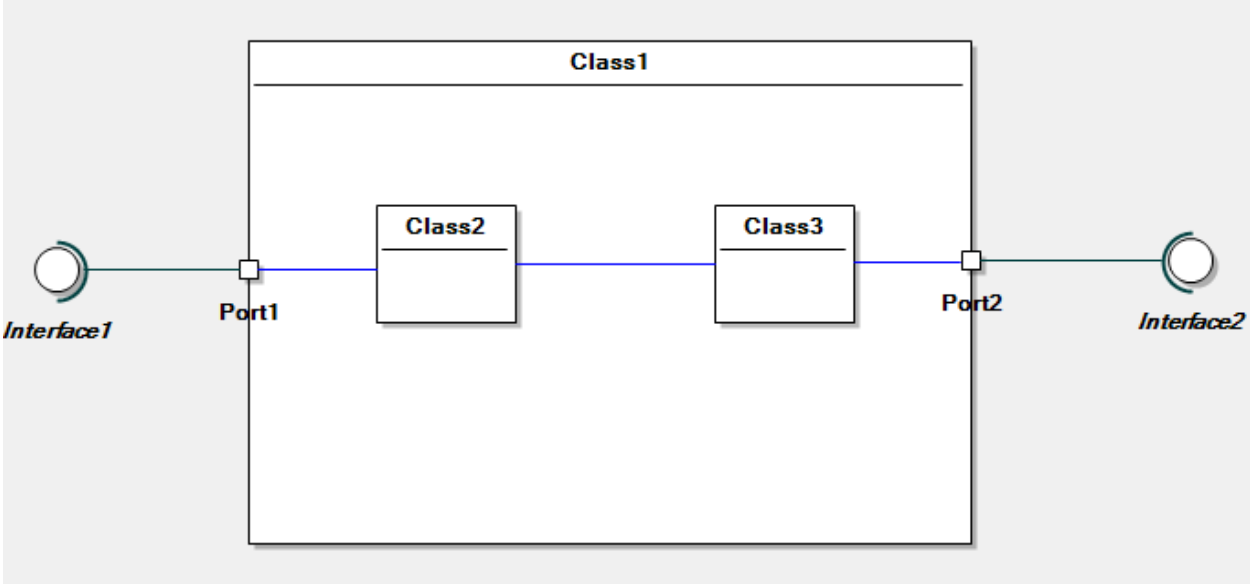


Рисунок 16 - Диаграмма композитной структуры, созданная в среде Borland Together

Диаграммы автомата

Диаграмма автомата, State Machine diagram (диаграмма конечного автомата, диаграмма состояний) — диаграмма, на которой представлен конечный автомат с простыми состояниями, переходами и композитными состояниями, как показано на рисунке 17.

Конечный автомат (State machine) — спецификация последовательности состояний, через которые проходит объект или взаимодействие в ответ на события своей жизни, а также ответные действия объекта на эти события. Конечный автомат прикреплён к исходному элементу (классу, кооперации или методу) и служит для определения поведения его экземпляров.

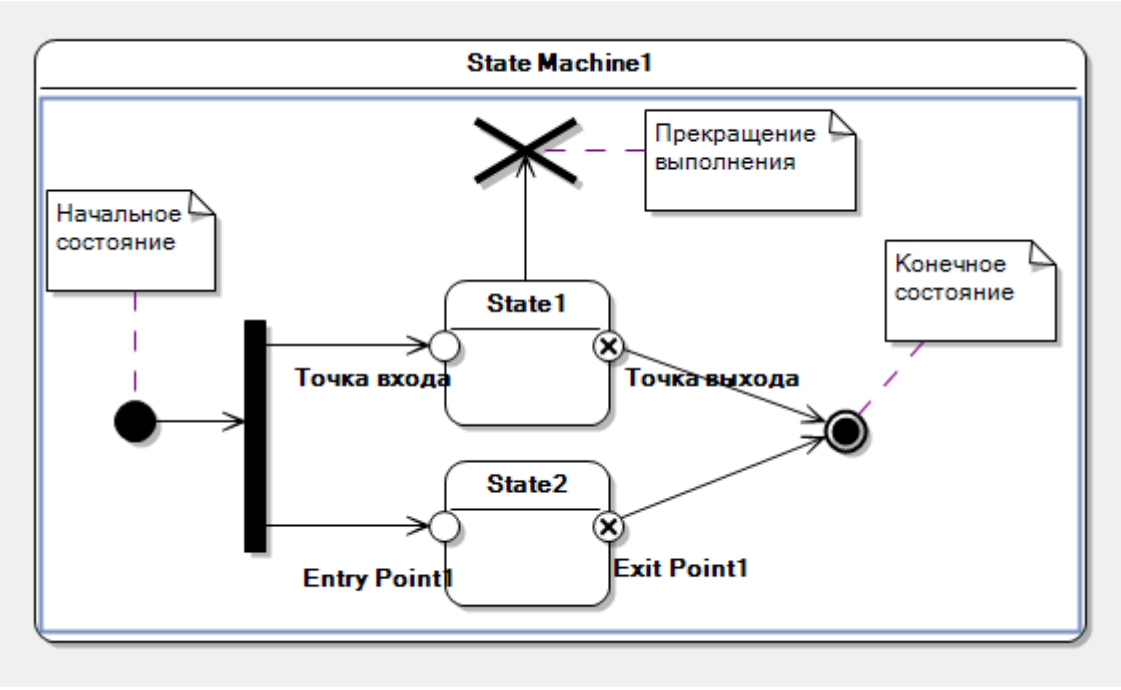


Рисунок 17 – Диаграмма автомата, созданная в среде Borland Together

Если в системе присутствуют объекты со сложным поведением, то строят диаграммы состояний. Построение диаграмм состояний может оказать следующее воздействие на описание классов:

-события могут отображаться в операции класса;

-особенности конкретных состояний могут повлиять на детали выполнения операций;

-описание состояний и переходов может помочь при определении атрибутов класса.

Каждая диаграмма состояний в UML описывает все возможные состояния одного экземпляра определенного класса и возможные последовательности его переходов из одного состояния в другое, то есть моделирует все изменения состояний объекта как его реакцию на внешние воздействия.

Диаграммы состояний чаще всего используются для описания поведения отдельных объектов, но также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

Главное предназначение этой диаграммы — описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

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

Действие рисуют вдоль линии перехода после имени события, его изображению предшествует наклонная (косая) черта.

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

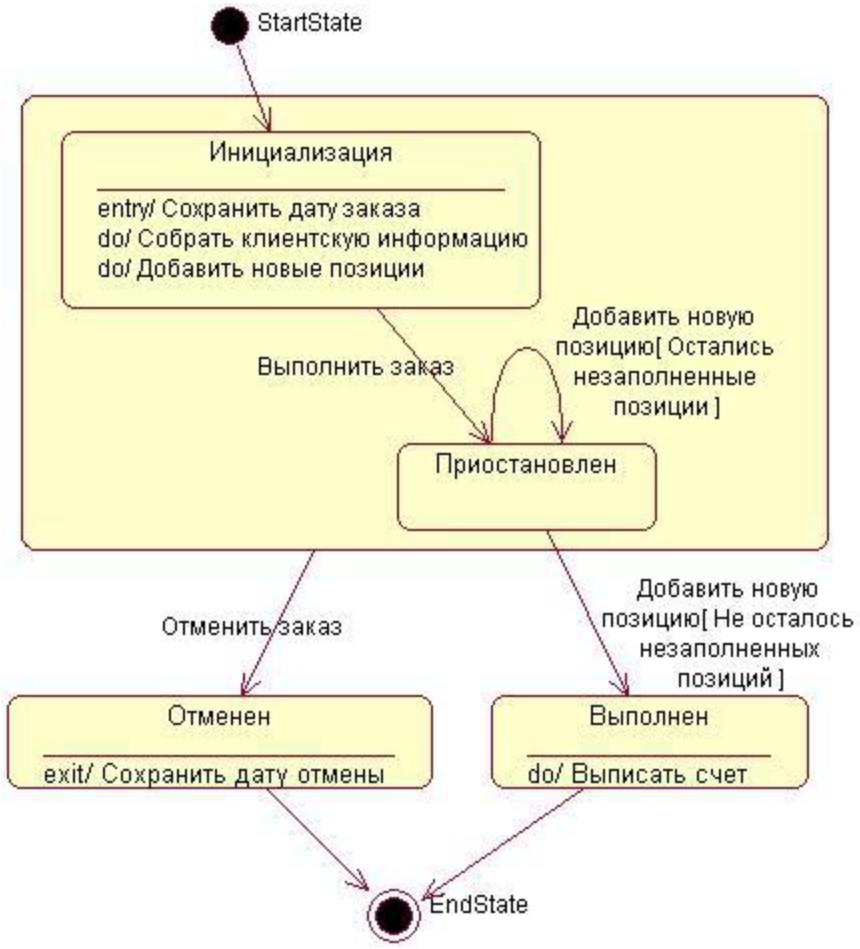


Рисунок 18 – Диаграмма состояний, созданная в среде Rational Rose

Для группировки классов, обладающих некоторой общностью, применяются пакеты. Пакет – общий механизм для организации элементов модели в группы. Каждый пакет – это группа элементов

модели, иногда сопровождаемая диаграммами, поясняющими структуру группы. Каждый элемент модели может входить только в один пакет. *Диаграммы пакетов* отображают зависимости между пакетами, возникающие, если элемент одного пакета зависит от элемента другого.

Пакеты также используются для представления подсистем. Подсистема – это комбинация пакета (поскольку она включает некоторое множество классов) и класса (поскольку она обладает поведением, т.е. реализует набор операций, которые определены в ее интерфейсах). Связь между подсистемой и интерфейсом называется связью реализации.

Жёсткого разделения между разными структурными диаграммами не проводится, поэтому данное название предлагается исключительно для удобства и не имеет семантического значения (пакеты и диаграммы пакетов могут присутствовать на других структурных диаграммах).

1.3.4 Реализация

Основная задача процесса реализации – создание системы в виде компонентов – исходных текстов программ, сценариев, двоичных файлов, исполняемых модулей и т.д. На этом этапе создается модель реализации, которая описывает то, как реализуются элементы модели проектирования, какие классы будут включены в конкретные компоненты. Данная модель описывает способ организации этих компонентов в соответствии с механизмами структурирования и разбиения на модули, принятыми в выбранной среде программирования и представляется диаграммой компонентов

Диаграмма компонентов

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Общий вид структуры информационной системы в виде диаграммы компонентов показан на рисунках 19 и 20. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ или реализации интерфейсов.

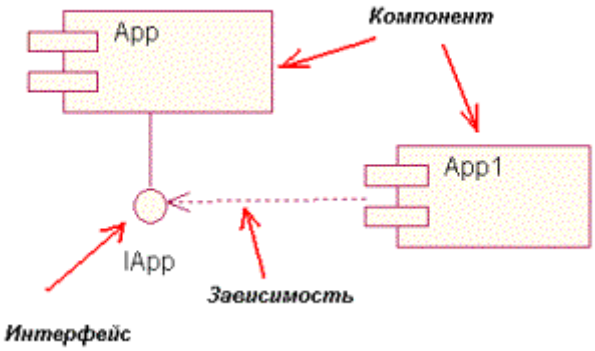


Рисунок 19 - Пример диаграммы компонентов на UML 1.5, созданной в среде Rational Rose

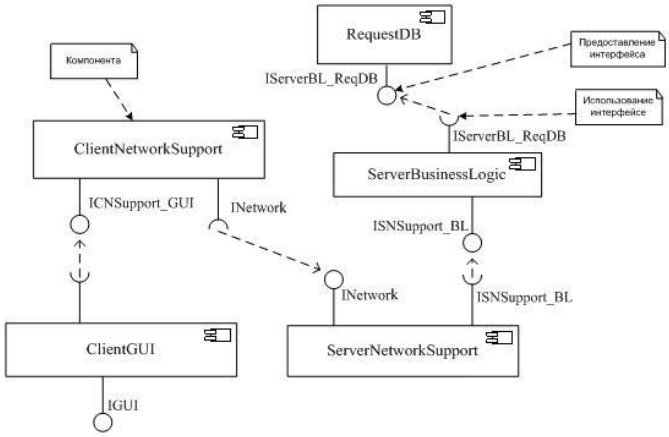


Рисунок 19 - Пример диаграммы компонентов на UML 2.0, созданной в среде Borland Together

Диаграмма развертывания

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Если разрабатывается программа, выполняющаяся локально на компьютере пользователя и не использующая периферийных устройств и ресурсов, то в разработке дополнительных диаграмм нет необходимости. При разработке же корпоративных приложений наличие таких диаграмм может быть крайне полезным для решения задач рационального размещения компонентов в целях эффективного использования распределенных вычислительных и коммуникационных ресурсов сети, обеспечения безопасности и других.

Для представления общей конфигурации и топологии распределенной программной системы в UML предназначены диаграммы развертывания.

Диаграмма развёртывания, Deployment diagram — служит для моделирования работающих узлов (аппаратных средств) и артефактов, развёрнутых на них. В UML 2 на узлах разворачиваются артефакты (artifact), в то время как в UML 1 на узлах разворачивались компоненты. Между артефактом и логическим элементом (компонентом), который он реализует, устанавливается зависимость манифестации. Это самый простой тип диаграмм, предназначенный для моделирования распределения устройств в сети. Для отображения используется всего два варианта значков процессор и устройство вместе со связями между ними.

Разработка диаграммы развертывания начинается с идентификации всех аппаратных, механических и других типов устройств, которые необходимы для выполнения системой всех своих функций. В первую очередь специфицируются вычислительные узлы системы, обладающие памятью и/или процессором.

Один из возможных вариантов построения диаграммы развертывания, созданной в среде Borland Together, показан на рисунке 21.

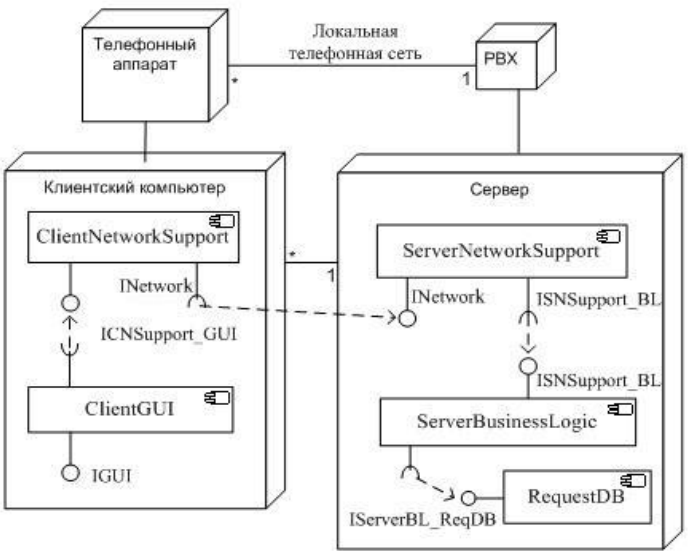


Рисунок 21 - Диаграмма развертывания, созданная в среде Borland Together

На диаграмме, показано каким образом компоненты телефонной службы приема заявок распределяются по аппаратной части системы.

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)				
6.1. Рекомендуемая литература				
6.1.1. Основная литература				
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л1.1	Игнатьев, С. А., Игнатьева, С. С.	Информационное обеспечение систем управления качеством: учебное пособие	Саратов: Саратовский государственный технический университет имени Ю.А. Гагарина, ЭБС АСВ, 2012	http://www.iprbookshop.ru/76484.html
Л1.2	Граничин О. Н., Кияев В. И.	Информационные технологии в управлении	Москва: Интернет-Университет Информационных Технологий	http://www.iprbookshop.ru/57379.html
Л1.3	Головицына М. В.	Информационные технологии в экономике	Москва: Интернет-Университет Информационных Технологий	http://www.iprbookshop.ru/52152.html
Л1.4	Малышева, Е. В.	Стратегическое планирование. Часть 1: учебное пособие	Новосибирск: Новосибирский государственный технический университет, 2010	http://www.iprbookshop.ru/45036.html
6.1.2. Дополнительная литература				
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л2.1	Гатина, Л. И.	Стратегическое планирование развития предприятии: учебно-методическое пособие	Казань: Казанский национальный исследовательский технологический университет, 2012	http://www.iprbookshop.ru/62291.html
Л2.2	Горбунов В. Л.	Бизнес-планирование	Москва: Интернет-Университет Информационных Технологий	http://www.iprbookshop.ru/56371.html
Л2.3	Силаенков, А. Н.	Информационное обеспечение и компьютерные технологии в научной и образовательной деятельности: учебное пособие	Омск: Омский государственный институт сервиса, Омский государственный технический университет, 2014	http://www.iprbookshop.ru/26682.html
6.1.3. Методические разработки				
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л3.1	Глазкова, И. Ю., Ловянников, Д. Г.	Информационные технологии в бизнес-планировании: лабораторный практикум	Ставрополь: Северо-Кавказский федеральный университет, 2017	http://www.iprbookshop.ru/75574.html
6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"				

Э1	Стасышин В.М. Проектирование информационных систем и баз данных [Электронный ресурс]: учебное пособие/ Стасышин В.М.— Электрон. текстовые данные.— Новосибирск: Новосибирский государственный технический университет, 2012.— 100 с.— Режим доступа: http://www.iprbookshop.ru/45001.— ЭБС «IPRbooks»
Э2	Тараненко Л.Г. Информационное обеспечение потребностей региона [Электронный ресурс]: учебное пособие/ Тараненко Л.Г.— Электрон. текстовые данные.— Кемерово: Кемеровский государственный институт культуры, 2009.— 194 с.— Режим доступа: http://www.iprbookshop.ru/21974.— ЭБС «IPRbooks»
Э3	Стратегическое планирование развития строительной организации [Электронный ресурс]/ А.Н. Асаул [и др.].— Электрон. текстовые данные.— СПб.: Институт проблем экономического возрождения, Санкт-Петербургский государственный архитектурно-строительный университет, 2009.— 166 с.— Режим доступа: http://www.iprbookshop.ru/18214.— ЭБС «IPRbooks»

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Организация и планирование экспериментов» для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Организация и планирование экспериментов» содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии, направленность (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины – дать представление о перспективах развития методов осуществления информационного обеспечения стратегического планирования, изучить цели, задачи, методы и способы осуществления информационного обеспечения, сформировать умения использовать методы информационного обеспечения на практике.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

В результате освоения данной дисциплины формируется следующая компетенция у обучающегося:

ПК-3.1: Адаптирует бизнес-процессы заказчика к возможностям информационной системы

Изучив данный курс, студент должен:

Знать:

методику априорного ранжирования факторов, метод наименьших квадратов, методы математического моделирования, методы оптимизации параметров иметь представление о математических моделях технических систем и о применении методов математического моделирования для исследования технических объектов.

Уметь:

самостоятельно работать с учебной, справочной и учебно-методической литературой; использовать методику априорного ранжирования факторов, применять метод наименьших квадратов, методы оптимизации параметров и методы математического моделирования; применять численные методы для решения задач с использованием прикладных математических пакетов.

Владеть:

учебной и учебно-методической литературой; навыками проведения экспериментальных исследований; навыками обработки и анализа результатов эксперимента; методом математического моделирования

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическое занятие 1

Разработка диаграмм унифицированного языка моделирования

Применение UML 2.0 позволяет разделить проблему моделирования сложной системы на составные части с помощью четырех представлений:

-статическое структурное представление модели описывает структурные аспекты системы, например, с помощью диаграммы классов;

-представление взаимодействия используется для моделирования последовательностей действий и коммуникаций, описывающих кооперацию взаимодействующих экземпляров;

-представление деятельности используется для создания моделей, описывающих поток «деятельностей» в системе;

-представление в виде конечного автомата используется для описания поведения системы в терминах состояний и переходов между ними.

Эти представления не являются полностью ортогональными: концепции, используемые в одном из них, часто зависят от концепций, применяемых в другом. Так, классификаторы участников взаимодействия должны быть определены в статической структурной модели. Такие зависимости определяются в метамодели UML, и инструментальные средства могут их задействовать для определения согласованности информации во всех представлениях системы.

Графические обозначения отдельных элементов моделей будут представлены при создании диаграмм в дальнейшем. Рассмотрим краткую характеристику диаграмм UML 2.0.

Структурные диаграммы

Диаграмма классов(Class diagram) — статическая структурная диаграмма, описывающая структуру системы, она демонстрирует классы системы, их атрибуты, методы и зависимости между классами.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

-концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;

-точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;

-точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).

Диаграмма компонентов(Component diagram) — статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п. Диаграмма компонента показывает структурные отношения между компонентами будущей информационной системы. В UML 2.0 компоненты являются автономными инкапсулированными единицами (unites) внутри системы или подсистемы, которые обеспечивают один или несколько интерфейсов. Поэтому диаграмма компонента позволяет архитектору убедиться в том, что компоненты реализуют заданную функциональность системы.

Диаграмма композитной/составной структуры (Composite structure diagram) — статическая структурная диаграмма, демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса. Подвидом диаграмм композитной структуры являются *диаграммы кооперации* (Collaboration diagram, введены в UML 2.0), которые показывают роли и взаимодействие классов в рамках кооперации. Кооперации удобны при моделировании шаблонов проектирования. Диаграммы композитной структуры могут использоваться совместно с диаграммами классов.

Диаграмма развёртывания(Deployment diagram) — служит для моделирования работающих узлов(аппаратных средств) и артефактов, развёрнутых на них. В UML 2 на узлах разворачиваются артефакты (*artifact*), в то время как в UML 1 на узлах разворачивались компоненты. Между артефактом и логическим элементом (компонентом), который он реализует, устанавливается зависимость манифестации.

Диаграмма объектов (Object diagram) — демонстрирует полный или частичный снимок моделируемой системы в заданный момент времени. На диаграмме объектов отображаются экземпляры классов (объекты) системы с указанием текущих значений их атрибутов и связей между объектами.

Диаграмма пакетов(Package diagram) — структурная диаграмма, основным содержанием которой являются пакеты и отношения между ними. Диаграммы пакетов служат, в первую очередь, для организации элементов в группы по какому-либо признаку с целью упрощения структуры и организации работы с моделью системы.

Диаграммы поведения

Диаграмма деятельности(Activity diagram) — диаграмма, на которой показано разложение некоторой *деятельности* на её составные части. Под деятельностью (англ. *activity*) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных *действий* (англ. *action*), соединённых между собой потоками, которые идут от выходов одного узла ко входам другого. Диаграммы деятельности используются при моделировании бизнес-процессов,

технологических процессов, последовательных и параллельных вычислений. Аналогом диаграмм деятельности являются схемы алгоритмов по ГОСТ 19.701-90.

Диаграмма автомата (State Machine diagram, *диаграмма конечного автомата*, *диаграмма состояний*) — диаграмма, на которой представлен *конечный автомат* с простыми состояниями, переходами и композитными состояниями. Конечный автомат — спецификация последовательности состояний, через которые проходит объект или взаимодействие в ответ на события своей жизни, а также ответные действия объекта на эти события. Конечный автомат прикреплён к исходному элементу (классу, кооперации или методу) и служит для определения поведения его экземпляров.

Диаграмма вариантов использования(Use case diagram) —представляет собой отражение действующих лиц (актантов), которые взаимодействуют с системой, и реакцию программных объектов на их действия. Актантами могут быть как пользователи, так и внешние агенты, которым необходимо передать или получить от них информацию. Значок варианта использования отражает реакцию системы на внешнее воздействие и показывает, что должно быть сделано для актанта. Основная задача — представлять собой единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.

Диаграммы взаимодействия

Диаграмма последовательности (Sequence diagram) — диаграмма, на которой изображено упорядоченное во времени взаимодействие объектов. В частности, на ней изображаются участвующие во взаимодействии объекты и последовательность сообщений, которыми они обмениваются. Диаграмма последовательности показывает хронологическую последовательность сообщений между объектами во взаимодействии. Она состоит из нескольких участников, таких как агенты, системы или подсистемы, классы и компоненты, представленные линиями жизни (lifelines), а также сообщения, которыми они обмениваются при взаимодействии.

Диаграмма коммуникации (Communication diagram, в UML 1.x — *диаграмма кооперации*, *collaboration diagram*) — диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. В отличие от диаграммы последовательности, на диаграмме коммуникации явно указываются отношения между элементами (объектами), а время как отдельное измерение не используется (применяются порядковые номера вызовов). Диаграмма коммуникации показывает поток сообщений между объектами и то, как несколько объектов сотрудничают при выполнении общей задачи. Как и диаграмма последовательности (sequence diagram), диаграмма коммуникации тоже может моделировать динамическое поведение для варианта использования (use case). Однако диаграмма коммуникации больше нацелена на показ того, как происходит координация, чем на хронометрирование последовательности.

Диаграммы коммуникации и последовательности транзитивны, выражают взаимодействие, но показывают его различными способами и с достаточной степенью точности могут быть преобразованы одна в другую.

Примечание.

По причине того, что диаграммы коммуникации и последовательности являются разными взглядами на одни и те же процессы, Rational Rose позволяет создавать из диаграммы коммуникации диаграмму последовательности и наоборот, а также производит автоматическую синхронизацию этих диаграмм.

Диаграмма обзора взаимодействия (Interaction overview diagram) — разновидность диаграммы деятельности, включающая фрагменты диаграммы последовательности и конструкции потока управления. Этот тип диаграмм включает в себя диаграммы последовательностей действий и диаграммы сотрудничества. Эти диаграммы позволяют с разных точек зрения рассмотреть взаимодействие объектов в создаваемой системе.

Диаграмма синхронизации(Timing diagram) — альтернативное представление диаграммы последовательности, явным образом показывающее изменения состояния на линии жизни с заданной шкалой времени. Эта диаграмма может быть полезна в приложениях реального времени.

Таким образом, выбранный разработчиком набор диаграмм позволяет создать практически любое представление о проектируемой системе.

Примечание.

Изображая диаграмму, воспользуйтесь следующими рекомендациями:

-дайте диаграмме имя, соответствующее ее назначению;

-расположите элементы так, чтобы свести к минимуму число пересечений;

-пространственно элементы расположите так, чтобы семантически близкие сущности располагались на диаграмме рядом;

-используйте примечания и цвет, чтобы привлечь внимание читателя к важным особенностям диаграммы.

Унифицированный процесс разработки программного обеспечения

Процесс проектирования ИСТ, кроме основных концепций и понятий, используемых при проектировании и реализации ИСТ, включает в себя технологию проектирования. Одной из развитых современных технологий является унифицированный процесс(Rational Unified Process,RUP). RUP – одна из лучших технологий разработки программного обеспечения, созданная в компании Rational Software, входящей в состав IBM. Унифицированный процесс позволяет создавать сложные программные системы, основываясь на индустриальных методах разработки [2, 3].

Вся разработка информационной системы (ИС) рассматривается в RUP как процесс создания артефактов. Любой результат работы проекта, будь то исходные тексты, объектные модули, документы, передаваемые пользователю, модели – это подклассы всех артефактов проекта.

Одним из интереснейших классов артефактов проекта являются модели, которые позволяют разработчикам определять, визуализировать, конструировать и документировать артефакты программных систем.

Модели позволяют рассмотреть будущую систему, ее объекты и их взаимодействие еще до вкладывания значительных средств в разработку, позволяют увидеть ее глазами будущих пользователей снаружи и разработчиков изнутри еще до создания первой строки исходного кода. Большинство моделей представляются диаграммами на унифицированном языке моделирования UML.

Основными моделями, создаваемыми в RUP, являются: модель вариантов использования, модель анализа, модель проектирования и модель реализации. Эти модели являются результатом основных работ процесса, к которым относятся: определение требований, анализ, проектирование и реализация. Рассмотрим содержание каждого из них.

1.3.1 Определение требований

Одним из важнейших этапов разработки ИС, согласно RUP, является этап определения требований, который заключается в сборе всех возможных пожеланий заказчика к работе системы. На данном этапе в ходе интервью с пользователями и изучения документов, аналитики должны собрать как можно больше требований к будущей системе, что не так просто, как кажется на первый взгляд. Позднее эти данные должны будут систематизированы и структурированы. Для того чтобы верно определить требования, разработчики должны понимать **контекст** (часть предметной области) в котором будет работать будущая система.

Определение контекста информационной системы

Для определения контекста ИСТ выполняется предпроектное обследование предметной области(области использования ИСТ). Для этого создаются модель предметной области и бизнес-модель, что является различными подходами к одному и тому же вопросу. Часто создается что-то одно: модель предметной области или бизнес-модель[2].

Отличия этих моделей в том, что модель предметной области описывает важные понятия, с которыми будет работать система и связи их между собой. Тогда как бизнес-модель описывает бизнес-процессы (существующие или будущие), которые должна автоматизировать(поддерживать) система. Поэтому кроме определения бизнес-объектов, вовлеченных в процесс, эта модель определяет работников, их обязанности и действия, которые они должны выполнять.

Использование UML не ограничивается моделированием программного обеспечения. Его также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Для создания модели предметной области с помощью UML используется обычная диаграмма классов, однако для создания бизнес-модели ее уже явно недостаточно. В этом случае применяется диаграмма вариантов использования с использованием дополнительных значков, которые отражают сущность бизнес-процессов – это бизнес-актант, бизнес-прецедент, бизнес-сущность и бизнес-управление. Эта модель намного ближе к следующей модели, создаваемой в процессе разработки – модели анализа.

При моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений часто используются диаграммы деятельности.

На практике диаграммы деятельности применяются в основном двумя способами:

-для моделирования процессов;

-для моделирования операций.

В первом случае внимание фокусируется на деятельности с точки зрения действующих лиц, которые работают с системой. Важным здесь является применимость диаграмм деятельности для описания бизнес-процессов. В данном случае для построения диаграмм деятельности используется так называемая траектория объекта, или поток объекта(object flow). Суть его состоит в том, что на диаграмме кроме деятельности можно изобразить и объекты, относящиеся к деятельности. С помощью символа зависимости(пунктирная стрелка) эти объекты можно соотнести с той деятельностью или переходом, где они создаются, изменяются или уничтожаются. Траектория объекта позволяет показать объекты, относящиеся к деятельности, а также моменты переходов этих объектов из одного состояния в другое.

Рекомендации по построению диаграмм деятельности для моделирования процессов заключаются в следующем.

Моделируют бизнес-процессы в несколько этапов, первым из которых является разбиение их на подпроцессы. Подпроцессы, являющиеся "участками большого процесса", описать легче.

Дальше выделяют ключевые объекты (и создают для них дорожки), определяют предусловия и постусловия каждого процесса (т. е. его границы), описывают деятельности и переходы, отображают на диаграммах состояния ключевых объектов, в которые они переходят в ходе процесса.

В итоге создается не какая-то абстрактная диаграмма, а модель реального бизнес-процесса в реальной компании, занимающейся реальным бизнесом.

Пример детализации конкретного бизнес-процесса с помощью диаграммы деятельности, созданной в Rational Rose, показан на рисунке 1. На диаграмме отражена деятельность выдачи товара со склада [1].

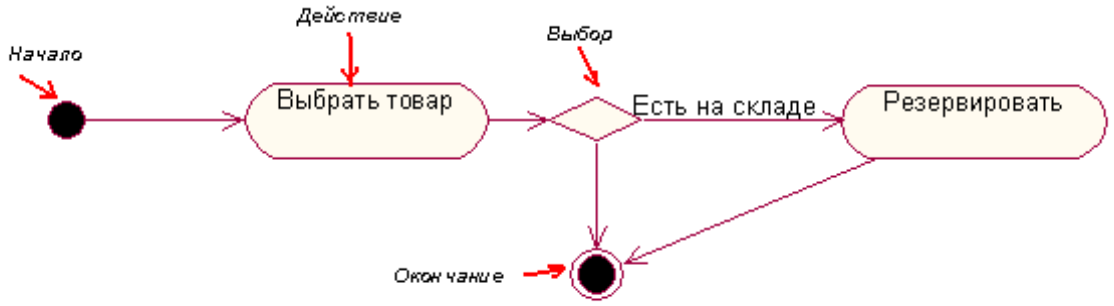


Рисунок 1 - Пример диаграммы активности, созданной в Rational Rose

Второй пример моделирования бизнес процесса оформление заказа в Интернет-магазине представлен на рисунке 2.

Рисунок 2 – Оформление заказа в Интернет-магазине

На рисунке 3 представлена диаграмма деятельности, выполненная в Borland Together. Здесь показана параметризованная деятельность с объектным узлом параметра, соединенным с контактами действий. Объектные узлы параметров размещаются на границе диаграммы, и ребра потока объектов соединяют их с контактами. Тип объекта, удерживаемого в объектном узле, обычно отображается в метке этого узла. В данном примере информация, используемая для заполнения заказа, предоставляется как входной параметр и передается в действие по вызову работы “Заполнение заказа”.

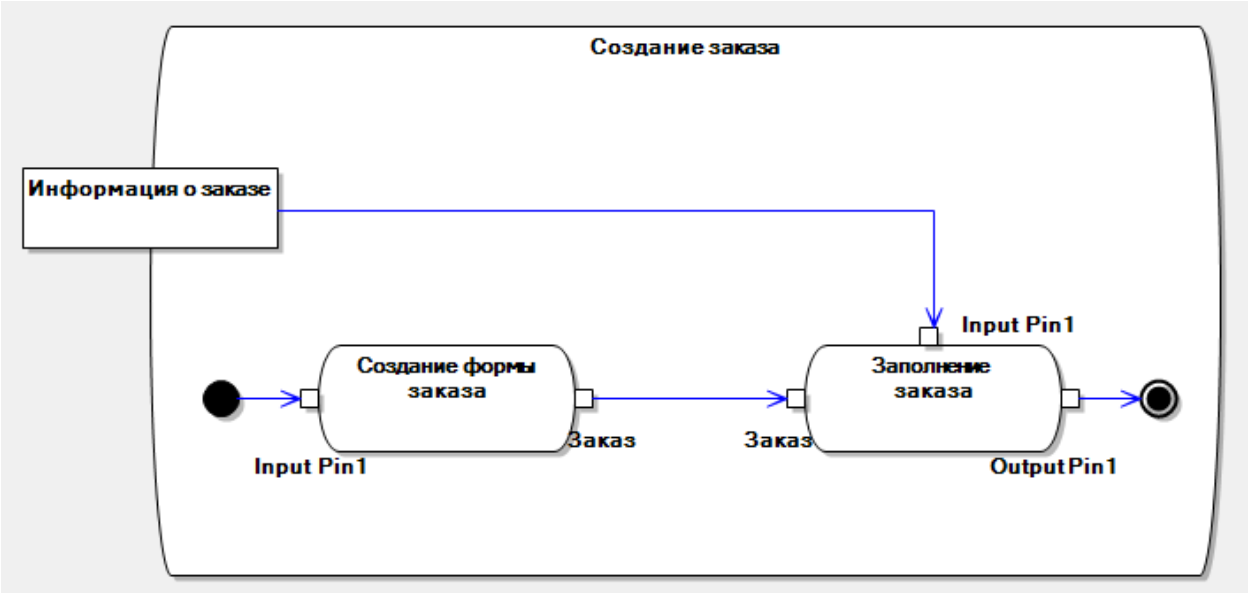


Рисунок 3 – Диаграмма активности, выполненная в Borland Together

Узлы управления в начале и в конце потока на рисунке 2 — это, соответственно, исходный и заключительный узлы. Когда вызывается деятельность “Создание заказа”, управляющий маркер помещается в начальный узел, а маркер данных с информацией о заказе — в объектный узел входного параметра. Управляющий маркер двигается от исходного узла к действию “Создание формы заказа”, которое начинает выполняться. Маркер данных передается от соответствующего параметра действию, вызывающему “Заполнение заказа”, которому приходится ждать до начала выполнения, пока “Создание формы заказа” не предоставит ему другие входные данные. После завершения действия “Заполнение заказа” управляющий маркер передается на конечный узел, деятельность завершается, а управление возвращается элементу, инициировавшему эту деятельность.

В случае моделирования операций диаграммы деятельности играют роль "продвинутых" блок-схем и применяются для подробного моделирования вычислений. На первое место при таком использовании выходят конструкции принятия решения, а также разделения и слияния потоков управления (синхронизации). Этот способ применяется при детализации вариантов использования и других процедур.

Рекомендации по построению диаграмм деятельности для моделирования операций заключаются в следующем.

Процесс построения диаграммы деятельности можно описать в виде последовательности таких действий:

1)Составление перечня деятельностей в системе

Как исходные данные для этой операции хорошо подходит список вариантов использования(или список операций). Дополняться диаграммой деятельности может каждый сценарий использования. Можно также попытаться описать связь между ними.

2)Определение зависимостей между деятельностями

Для каждой деятельности нужно найти деятельности, непосредственно предшествующие (и следующие за ней тоже), то есть деятельности, без выполнения которых поток управления не может перейти к данной деятельности.

3)Выделение параллельных потоков деятельностей

Выделяются деятельности, имеющие общих предшественников.

4)Определение условий переходов

Для этого формулируются выражения, которые могут принимать только два значения - "истинно" или "ложно", соответствующие альтернативным потокам управления.

5)Уточнение сложных деятельностей

Повторяя пункты 1-4 для каждой из деятельностей(при необходимости), можно уточнить сложные деятельности.

Таким образом, диаграммы деятельности в UML используются для моделирования потоков различного типа: потоков сигналов или данных, а также алгоритмических или процедурных потоков.

Пример диаграммы деятельности, выполненной в MS Office Visio, представлен на рисунке 4.

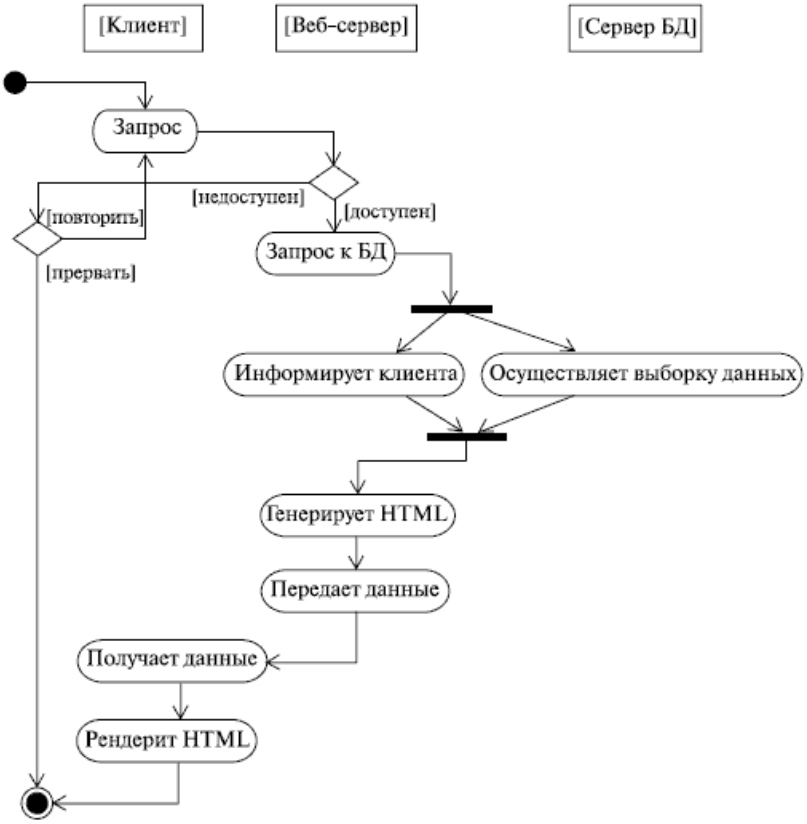


Рисунок 4 - Диаграмма деятельности, выполненная в MS Office Visio

На диаграмме показана работа с веб-приложением, решающим некую задачу в удаленной базе данных. Привлекает внимание расположение деятельностей на этой диаграмме: они как бы разбросаны по трем дорожкам, каждая из которых соответствует поведению одного из трех объектов - клиента, веб-

сервера и сервера баз данных. Благодаря этому легко определить, каким из объектов выполняется каждая из активностей, что очень упрощает ее восприятие.

Аналогия с дорожками действительно очень удачна. Именно таково официальное название элемента нотации UML, позволяющего указать распределение ролей на диаграмме деятельности.

Создавая диаграммы деятельности, необходимо учитывать, что они лишь моделируют срез некоторых динамических аспектов поведения системы. С помощью единственной диаграммы деятельности никогда не удастся охватить все динамические аспекты системы. Вместо этого следует использовать разные диаграммы деятельности для моделирования динамики рабочих процессов или отдельных операций.

Спецификация требований к информационной системе

Основным средством спецификации требований к проектируемой информационной системе в рамках RUP является модель вариантов использования. Главное назначение диаграммы вариантов использования заключается в формализации функциональных требований к системе. Основная задача — представить единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.

Модель варианта использования дает подробную информацию о поведении системы или приложения, которое разрабатывается. Она определяет требования к системе в терминах требующейся функциональности (вариантов использования) для достижения целей или для решения проблемы, определенной пользователем. Она же описывает окружение(агенты) и отношения между вариантами использования и агентами. Модель вариантов использования обычно включает в себя диаграммы вариантов использования и диаграммы действий, которые описывают то, как пользователи общаются с системой.

Цель варианта использования заключается в том, чтобы определить законченный аспект или фрагмент поведения некоторой сущности без раскрытия внутренней структуры этой сущности. В качестве такой сущности может выступать исходная система или любой другой элемент модели, который обладает собственным поведением, подобно подсистеме или классу в модели системы.

Каждый вариант использования соответствует отдельному сервису, который предоставляет моделируемую сущность или систему по запросу пользователя (актера), т. е. определяет способ применения этой сущности. Сервис, который инициализируется по запросу пользователя, представляет собой законченную последовательность действий. Это означает, что после того как система закончит обработку запроса пользователя, она должна возвратиться в исходное состояние, в котором готова к выполнению следующих запросов.

Примерами вариантов использования могут являться следующие действия: проверка состояния текущего счета клиента, оформление заказа на покупку товара, получение дополнительной информации о кредитоспособности клиента, отображение графической формы на экране монитора и другие действия.

Пример простейшей диаграммы вариантов использования “Заказ товара”, выполненной в Rational Rose, представлен на рисунке 5. На диаграмме показаны условные графические изображения главных элементов диаграммы – действующего лица(актанта) и варианта использования, а также связи между ними.

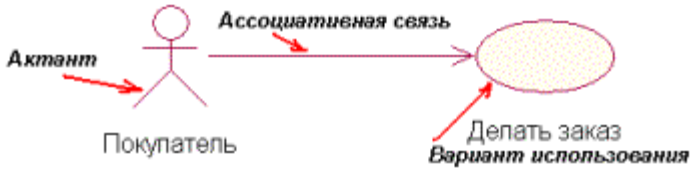


Рисунок 5 - Диаграмма вариантов использования,
выполненная в Rational Rose

Пример диаграммы вариантов использования, выполненной в Borland Together, показан на рисунке 6.

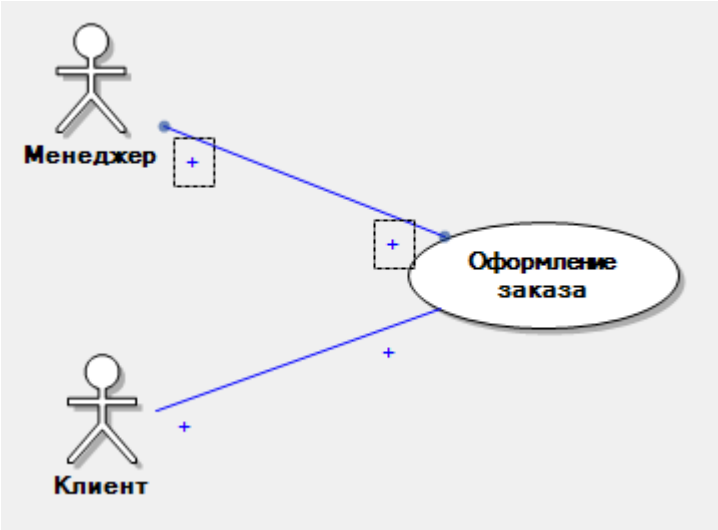


Рисунок 6 – Диаграмма вариантов использования, выполненная в Borland Together

Далее после создания диаграммы вариантов использования следует определить реализацию каждого варианта использования. Для этого применяются следующие способы:

-текстовое описание;

-описание алгоритма с помощью диаграмм деятельности;

-создание одной или несколько диаграмм взаимодействия.

Текстовое описание, соответствующее основной модели в рамках RUP, представляет собой:

-краткое описание;

-действующие лица;

-специальные требования;

-предпосылки;

-постусловия;

-точки расширения.

Например, рассмотрим вариант использования “Сделать предложение на аукционе”, диаграмма которого представлена на рисунке 7.

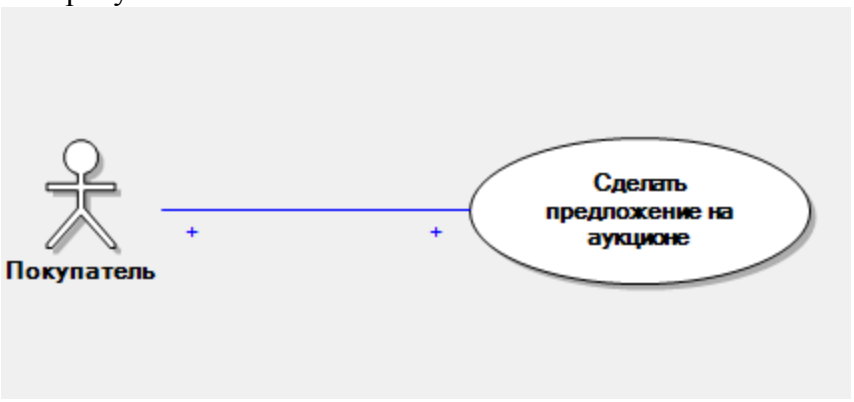


Рисунок 7 – Пример диаграммы варианта использования

В случае с приложением по ведению аукциона, речь может идти о представленном ниже потоке событий(последовательности, инициированной действующим лицом при подаче заявки в системе аукциона):

Основной поток:

-заявка (предложение цены): прецедент начинается в тот момент, когда покупатель предлагает свою цену на текущую позицию;

-ввод суммы: покупатель вводит сумму предложения. Система подтверждает, что сумма предложения превышает текущую ставку на значение, кратное шагу заявки для данной позиции;

-покупатель подтверждает заявку: покупатель подтверждает свое намерение разместить заявку;

-обработка заявки: система добавляет заявку к данной позиции;

-подтверждение заявки: система подтверждает наличие заявки путем отправки покупателю электронного сообщения. Продавец также уведомляется по электронной почте.

Альтернативные потоки операций могут описывать, что произойдет, если прием заявок будет прекращен до подачи предложения, если сумма заявки будет признана недействительной или покупатель не подтвердит подачу заявки.

Кроме текстового описания для детализации конкретного варианта использования(прецедента) можно построить диаграмму деятельности, правила построения которых аналогичны рассмотренных ранее.

Один из основных способов представления реализации варианта использования является создать одну или несколько диаграмм взаимодействия в форме диаграмм коммуникации или диаграмм последовательности, которые описывают один или несколько сценариев данного варианта использования. Этот способ в наибольшей степени соответствует идеологии UML и рекомендуется как основной и предпочтительный. Все эти операции выполняются на этапе анализа.

1.3.2 Анализ

После определения контекста, в котором будет работать система и требований к ней, наступает черед анализа полученных данных. В процессе анализа создается **аналитическая модель(модель анализа),** которая подводит разработчиков к архитектуре будущей системы. Аналитическая модель – это взгляд на систему изнутри, в отличие от модели вариантов использования, которая показывает, как система будет выглядеть снаружи.

Эта модель позволяет понять, как система должна быть спроектирована, какие в ней должны быть классы и как они должны взаимодействовать между собой. Основное ее назначение - определить направление реализации функциональности, выявленной на этапе сбора требований и сделать набросок архитектуры системы.

Модель анализа описывает логическую структуру системы и является фундаментом модели проектирования. Но в отличие от создаваемой в дальнейшем модели проектирования, модель анализа является в большей степени концептуальной моделью и только приближает разработчиков к классам реализации. Эта модель не должна иметь возможных противоречий, которые могут встретиться в модели вариантов использования.

Для построения аналитической модели выполняется анализ вариантов использования, который включает в себя:

-идентификацию классов, участвующих в реализации потоков событий;

-определение обязанностей классов;

-определение атрибутов и ассоциаций классов;

-унификацию классов анализа.

В потоках событий варианта использования выявляются классы трех типов:

-граничные классы, являющиеся посредниками при взаимодействии с внешними объектами;

-классы-сущности, представляющие собой основные абстракции (понятия) разрабатываемой системы;

-управляющие классы, обеспечивающие координацию поведения объектов в системе.

Классы анализа отражают функциональные требования к системе и моделируют объекты предметной области. Совокупность классов анализа представляет собой начальную концептуальную модель системы.

Для отображения модели анализа при помощи UML используется диаграмма классов со стереотипами (образцами поведения) «граничный класс», «сущность», «управление», а для детализации используются диаграммы взаимодействия, которые описывают взаимодействие групп объектов в различных условиях их поведения. Наиболее используемым типом таких диаграмм являются диаграммы коммуникации и последовательности.

Диаграмма коммуникации

Диаграмма коммуникации делает фокус на представлении группы взаимодействующих объектов и связей между ними, образующихся, если объекты общаются друг с другом посредством отсылки и приема сообщений. Также диаграммы коммуникаций подобны диаграммам объектов, но на них дополнительно могут быть показаны отсылаемые сообщения, причем допускается даже с указанием нумерации, описывающей порядок их следования во времени.

Диаграмма коммуникации используется для описания поведения системы как последовательности обмена сообщениями между элементами.

Основные сущности, используемые на диаграмме:

-роли, которые играют взаимодействующие элементы;

-объекты – экземпляры конкретных классов;

- связи - отношения, соединяющие взаимодействующие элементы.

Диаграмма коммуникации описывает поведение как взаимодействие, т. е. как протокол обмена сообщений между объектами.

Построение диаграммы коммуникации (кооперации) можно начинать сразу после построения диаграммы вариантов использования. В этом случае каждый из вариантов использования может быть специфицирован в виде отдельной диаграммы кооперации уровня спецификации.

Главная особенность диаграммы коммуникации (кооперации, сотрудничества) заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Прежде всего, на диаграмме коммуникации(кооперации, сотрудничества) в виде прямоугольников(окружностей) изображаются участвующие во взаимодействии объекты, содержащие имя объекта, его класс и, возможно, значения атрибутов. Далее должны быть изображены динамические связи - потоки сообщений. Они представляются в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений.

Пример диаграммы коммуникации (сотрудничества) показан на рисунке 8. Здесь показаны объект класса сущности “счет”, объект класса управления “обработчик” и объект граничного класса “интерфейс запроса на оплату”.

Стереотип «граничный класс» отображает класс, который взаимодействует с внешними актантами, «сущность» – отображает классы, которые являются хранилищами данных, а «управление» – классы, управляющие запросами к сущностям.

Линии, соединяющие объекты классов, отражают их взаимодействие.

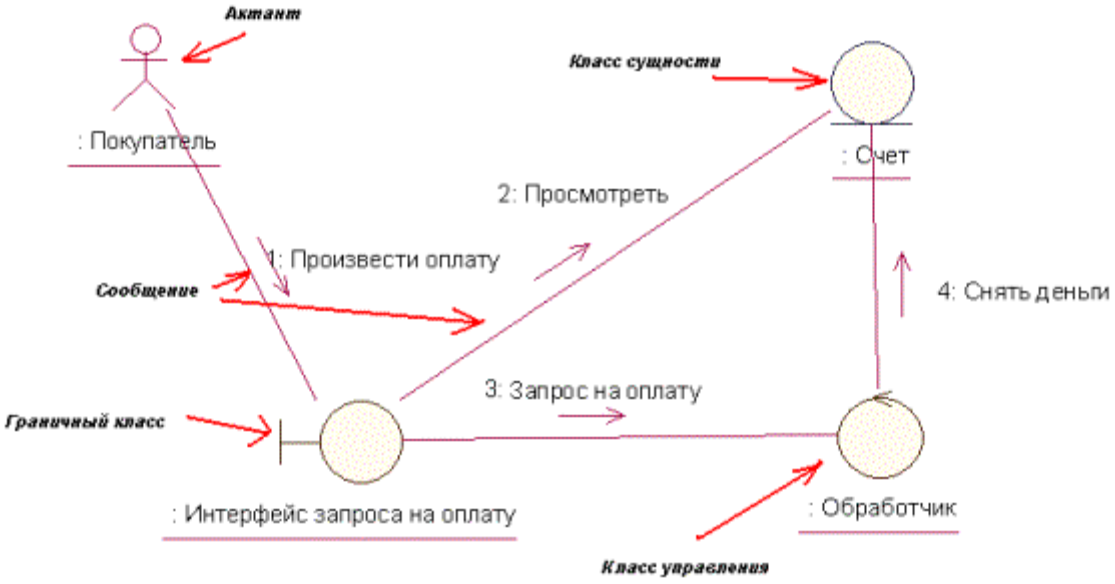


Рисунок 8 - Пример диаграммы коммуникации (сотрудничества)

Нумерация сообщений показывает их порядок, однако назначение диаграммы не в том, чтобы рассмотреть порядок обмена сообщениями, а в том, чтобы наглядно показать связи классов друг с другом.

При создании диаграммы коммуникации можно явно указать имена ассоциаций и ролей, которые играют объекты в данной ассоциации, как показано на диаграмме, изображенной на рисунке 9. Здесь показаны ассоциации “Продажа товара” и “Продажа компьютера”, а также роли “ клиент” и “менеджер”.

Рисунок 9 - Пример изображения ассоциаций и ролей

Диаграмма последовательности

Если акцентировать внимание на порядке взаимодействия, то другим его представлением будет диаграмма последовательности (Sequence).

Диаграмма последовательности - это диаграмма, чаще всего, описывающая один сценарий приложения. На диаграмме изображаются экземпляры объектов и сообщения, которыми они обмениваются в рамках одного варианта использования. Участники диаграммы именуются следующим образом: **имя: Класс**, где и имя, и класс являются не обязательными, но если используется класс, то присутствие двоеточия обязательно.

На диаграмме последовательности, каждый участник представлен вместе со своей линией жизни (lifeline), это вертикальная линия под объектом, вертикально упорядочивающая сообщения на странице. Важно: все сообщения на диаграмме следует читать сверху вниз. Каждая линия жизни имеет полосу активности (прямоугольники), которая показывает интервал активности каждого участника при взаимодействии.

Обозначение различных сообщений на диаграмме показано на рисунке 10.

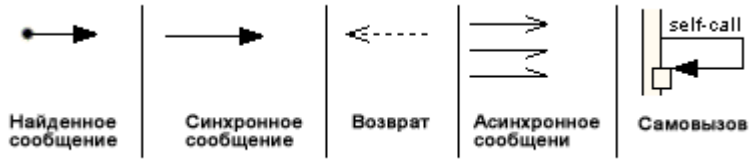


Рисунок 10 - Обозначение различных сообщений на диаграмме

У первого сообщения нет участника, пославшего его, поскольку оно приходит от неизвестного источника. Такое сообщение называется найденным сообщением (found message). Отправитель или получатель сообщения может находиться за пределами диаграммы коммуникации, и в этом случае используют входной и выходной шлюзы.

Сообщения, которыми обмениваются элементы, могут быть синхронными или асинхронными, что отражается в нотации стрелочек. Синхронные (synchronous message) - требующие возврата ответа, а асинхронные (asynchronous message) – ответа не требуют(вызывающий объект может продолжать работу). На диаграмме синхронные вызовы обозначаются закрашенными стрелочками, асинхронные – не закрашенными или половинными стрелочками.

Обратной пунктирной стрелкой показывается возврат ответа на сообщение(если сообщение является синхронным). Лучше применять изображение возврата только в тех случаях, когда это поможет лучше понять устройство взаимодействия. Во всех остальных случаях, стоит опускать изображения возвратов, т.к. они будут вносить некоторую неразбериху. Просто, при использовании синхронного сообщения, стоит помнить, что у него всегда есть возврат.

Если элемент диаграммы связан сам с собой, то такая связь называется рефлексивной(самовызов).

Если в сообщении требуется передать параметры, то они указываются в скобках через запятую, с указанием типа параметра (messageText(text : string)).

Для того чтобы задать порядок следования сообщений, используют десятичную нумерацию.

Пример диаграммы последовательности показан на рисунке 11. На ней представлены два объекта и все возможные виды сообщений.

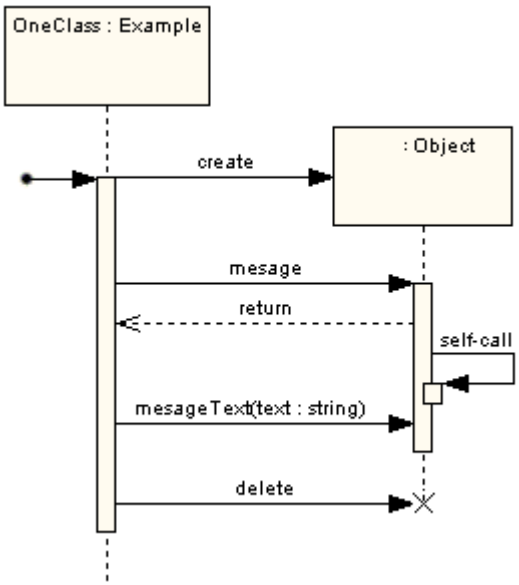


Рисунок 11 - Виды сообщений, которыми обмениваются объекты

Диаграмма последовательности, соответствующая диаграмме сотрудничества, показанной на рисунке 8, представлена на рисунке 12. Виды сообщений, которыми обмениваются объекты, аналогичны сообщениям диаграммы коммуникации, изображенным на рисунке 10.

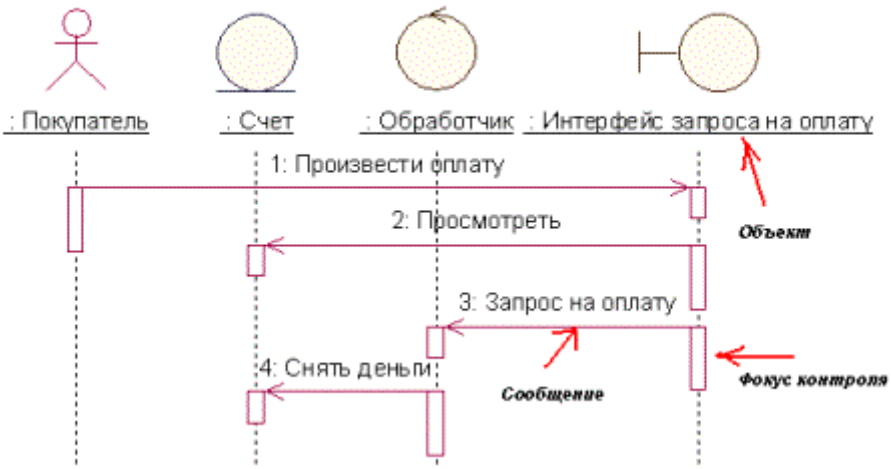


Рисунок 12 - Пример диаграммы последовательности действий

Решение о том какую из двух диаграмм нужно создавать первой, зависит от предпочтений конкретного разработчика. Поскольку эти диаграммы являются отображением одного и того же процесса, то и та и другая позволяют отразить взаимодействие между объектами.

Диаграммы коммуникации и последовательности транзитивны, выражают взаимодействие, но показывают его различными способами и с достаточной степенью точности могут быть преобразованы одна в другую.

При использовании такого инструмента для создания моделей как Rational Rose, эти два вида диаграмм могут быть созданы друг из друга автоматически [5].

В некоторых случаях могут строиться диаграммы обзора взаимодействия и диаграммы синхронизации.

1.3.3 Проектирование

Следующим этапом в процессе создания системы будет проектирование, в ходе которого на основании моделей, созданных ранее, создается **модель проектирования**. Эта модель отражает физическую реализации системы и описывает создаваемый продукт на уровне классов и компонентов. В отличие от модели анализа, модель проектирования имеет явно выраженную зависимость от условий реализации, применяемых языков программирования и компонентов.

Модель проектирования, используя различные диаграммы (в том числе диаграммы последовательности, машины состояний, компонента и размещения), подробно описывает, как устроено приложение и как оно будет реализовываться. Она также описывает структурные компоненты программ и технологий, например, обеспечивающих персистентность, распределение, безопасность и доступ к данным.

Для максимально точного понимания архитектуры системы, эта модель должна быть максимально формализована, и поддерживаться в актуальном состоянии на протяжении всего жизненного цикла разработки системы.

В RUP проектирование концентрируется вокруг определения архитектуры системы, а для систем с большой долей программного обеспечения - вокруг архитектуры программного обеспечения. Использование компонентных архитектур - один из шести наилучших подходов к разработке программ, инкорпорированных в RUP, рекомендует уделять больше времени на разработку и сопровождение архитектур. Время, затраченное на эти усилия, сокращает риски, связанные с ненадежными и негибкими системами.

Для создания модели проектирования используются целый набор UML диаграмм: диаграммы классов, диаграммы композитной структуры(кооперации), диаграммы взаимодействия, диаграммы активности. Основной является диаграмма классов.

Диаграмма классов

Диаграмма классов является основным типом диаграммы статической структуры. Она описывает структуру системы, показывая её классы, их атрибуты и операторы, и также взаимосвязи этих классов.

Каждый класс имеет имя, размещенное в верхнем блоке прямоугольника, изображающего класс. Для атрибутов и операций в элементах отводится отдельный блок. Каждый блок разделяется горизонтальной чертой.

Для атрибутов и операций применяются спецификаторы доступа. Спецификатора доступа языка C++ (public, private, protected) в UML отображаются символами + (public), - (private), # (protected), которые ставятся перед именем атрибута/операции. Также возможен вариант с ключевыми словами public, private, protected. Значение спецификаторов доступа: public - поля/методы класса видны снаружи класса. Т.е. к ним могут получать доступ объекты класса. private - поля/методы класса видны только внутри определения класса. protected - поля/методы класса видны в определении самого класса и в определениях производных классов.

Между классами существуют различные виды взаимодействия (или связи): один класс может быть производным другого, третий может содержать объект четвёртого в виде поля и т.д. Для различных видов взаимодействия в UML есть специальные названия.

Первый вид взаимодействия - ассоциация(association).

Ассоциация – это семейство связей двух и более классов. Обычно ассоциация возникает, когда один класс вызывает метод другого или если при вызове метода в качестве аргумента передаётся объект другого класса. Иногда при ассоциации показывают направленность (если это имеет значение).

Частным случаем ассоциации является связь – простая взаимосвязь между объектами. Она представляется линией соединяющей два или более объектных блока. Она встречается на диаграммах классов или объектов.

Всего существует пять типов ассоциации. Но наиболее распространены два: двунаправленная и однонаправленная ассоциации.

Сообщение направленная ассоциация(Message/Directed Association) используется, когда один класс “общается” с другим при создании экземпляра класса. Экземпляр класса — это описание конкретного объекта в памяти. Класс описывает свойства и методы, которые будет доступны у объекта, построенного по описанию, заложенному в класс. Экземпляры используют для представления конкретных сущностей реального мира.

Графически направленная ассоциация представляется в виде стрелочки направленной к “вызываемому” классу.

Частными вариантами ассоциации являются: агрегация и композиция.

Агрегация(быть частью) применяется, когда один класс должен быть контейнером других классов. Причем время существования содержащихся классов никак не зависит от времени существования класса контейнера. Графически агрегация представляется пустым ромбиком на блоке класса и линией, идущей от этого ромбика к содержащемуся классу.

Композиция - еще один случай ассоциации, но более строгий. В отличии от агрегации, композиция имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер будет уничтожен, то всё его содержимое будет уничтожено также. Графически представляется, как и агрегация, но с закрашенным ромбиком.

Различие между этими двумя видами ассоциации состоит в том, что композиция может быть частью одного и только одного целого, в то время как агрегация может быть частью нескольких объектов.

Еще одной разновидностью связи является **генерализация** (обобщение). Генерализация показывает, что один из двух связанных классов (*подтип*), является более частной формой другого (*супертип*), который называется обобщением первого. Графически генерализация представляется линией с пустым треугольником у супертипа.

Последнее отношение, которое мы рассмотрим, будет **реализация**(realization). Данная связь показывает отношение: класс - объект. На диаграмме реализация показывается пунктирной линией и не закрашенной стрелочкой.

Одной из важнейших характеристик взаимодействия является кратность(multiplicity) роли ассоциации. Кратностью роли ассоциации называется характеристика, указывающая, сколько объектов класса с данной ролью может или должно участвовать в каждом экземпляре ассоциации.

Наиболее распространенным способом задания кратности роли ассоциации является указание конкретного числа или диапазона. Например, указание “1” говорит о том, что все объекты класса с данной ролью должны участвовать в некотором экземпляре данной ассоциации, причем в каждом экземпляре ассоциации может участвовать ровно один объект класса с данной ролью. Указание диапазона “0..1” говорит о том, что не все объекты класса с данной ролью обязаны участвовать в каком-либо экземпляре данной ассоциации, но в каждом экземпляре ассоциации может участвовать только один объект. Аналогично, указание диапазона “1..*” говорит, что все объекты класса с данной ролью должны участвовать в некотором экземпляре данной ассоциации, и в каждом экземпляре ассоциации должен участвовать хотя бы один объект (верхняя граница не задана).

Пример диаграммы классов, на которой показаны все возможные варианты связей, представлен на рисунке 13.

Проектирование классов на данном этапе включает следующие действия:

-детализация проектных классов;

-уточнение операций и атрибутов;

-уточнение связей между классами;

-моделирование состояний для классов.

Детализация проектных классов, определенных в процессе анализа, уточнение атрибутов классов заключается в следующем:

-задается тип атрибута и значение по умолчанию (необязательно);

-задается видимость атрибутов: public, private или protected;

-при необходимости определяются производные (вычисляемые) атрибуты.

Обязанности классов, определенные в процессе анализа и документированные в виде «операций анализа», преобразуются в операции, которые будут реализованы в коде. При этом:

-каждой операции присваивается краткое имя, характеризующее ее результат;

-определяется полная сигнатура операции;

-создается краткое описание операции, включая смысл всех ее параметров;

-определяется видимость операции: public, private или protected;

-определяется область действия операции: операция объекта или операция класса.

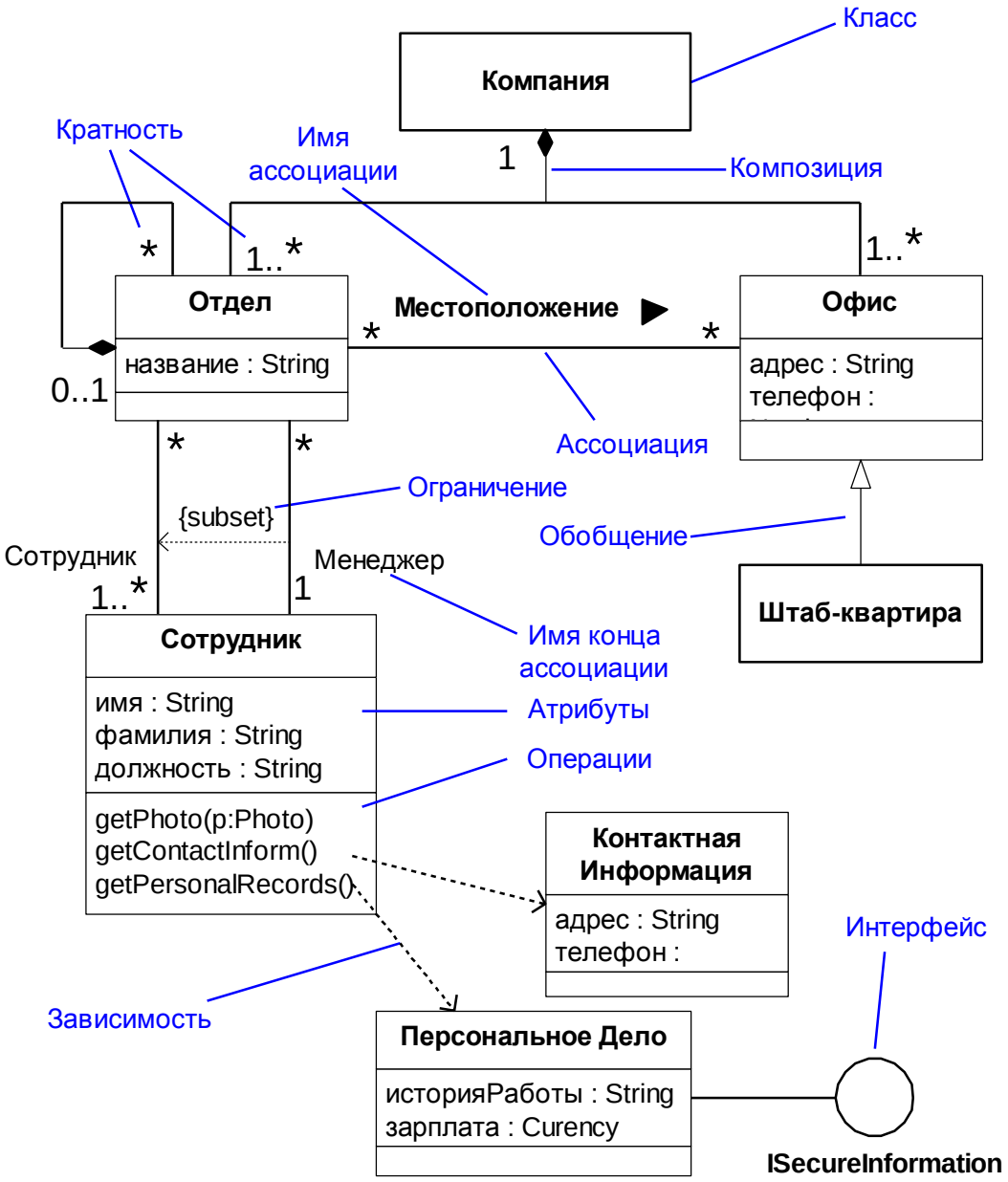


Рисунок 13 - Пример диаграммы классов

В процессе проектирования связи между классами подлежат уточнению. Ассоциации между граничными и управляющими классами преобразуются в зависимости. Агрегации, обладающие свойствами композиции, преобразуются в связи композиции. Связи обобщения могут преобразовываться в ситуациях с так называемой метаморфозой подтипов, когда объект суперкласса может менять свой подтип

Например, для бизнес-процесса – продажа товаров по заказу(каталогу), можно выделить классы Заказ и Клиент, который может представляться как юридическим, так и физическим лицом. Каждый класс имеет определенные атрибуты(свойства). Заказ выполняется в определенный день. Клиент имеет имя и т.д.

Пример диаграммы классов, созданной в среде Rational Rose, показан на рисунке 14.

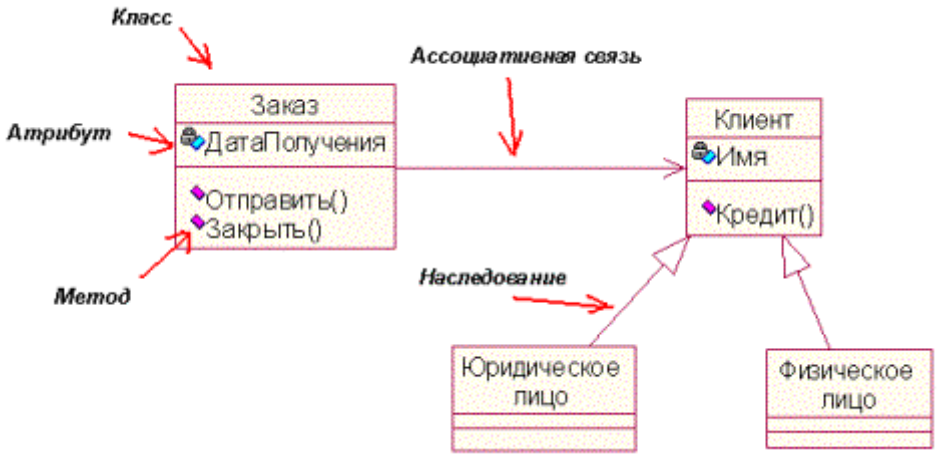


Рисунок 14 - Пример диаграммы классов, созданной в среде Rational Rose

Из диаграммы видно, что между классами “Заказ” и “Клиент” имеет место связь однонаправленная ассоциация.

Из диаграммы видно, что базовый класс “Клиент” имеет два производных класса “Юридическое лицо” и “Физическое лицо”, соединенных с базовым классом связью типа “Обобщение”, реализующей принцип наследования.

Другой вариант диаграммы классов для того же бизнес-процесса, созданной в среде Borland Together, показан на рисунке 15. На этой диаграмме классы имеют более полный набор атрибутов и операций. Однако класса Клиент не является базовым. Наследования классов на данной диаграмме не предусмотрено.

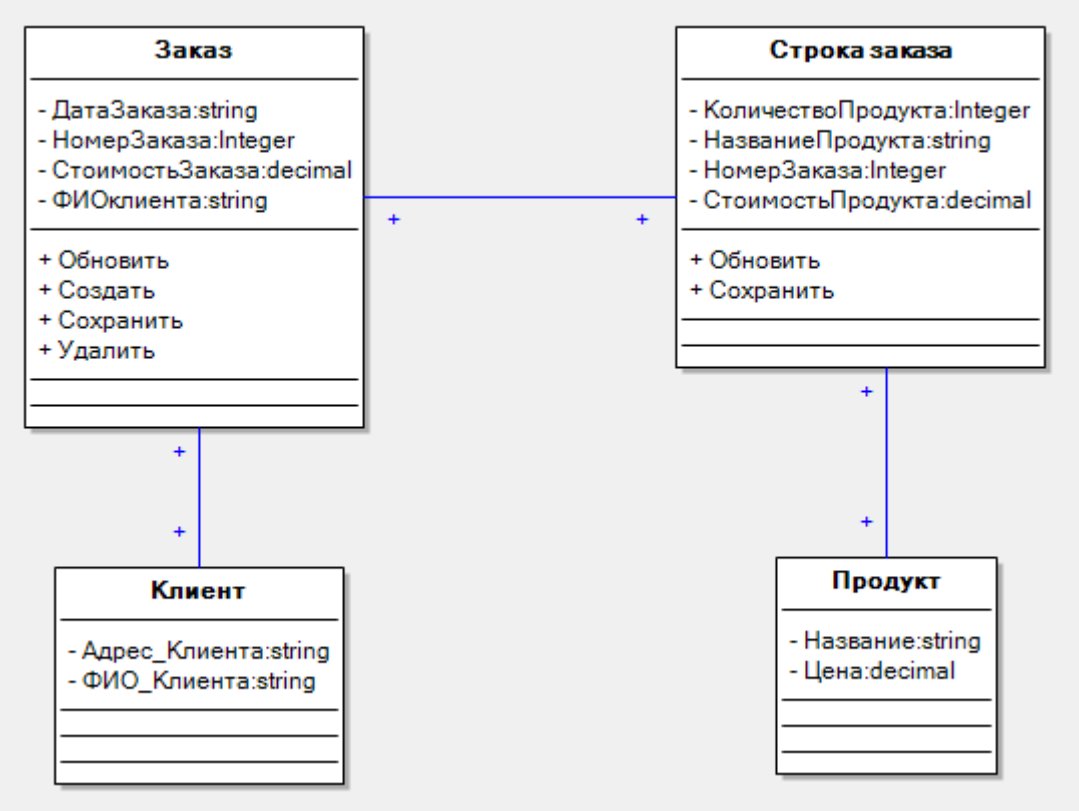


Рисунок 15 - Диаграмма классов, созданная в среде Borland Together

Основные правила построения диаграмм классов

В UML необязательно расписывать все детали классов. Это будет сделано при написании кода на конкретном языке (в нашем случае - C++). В UML-диаграмме можно опускать ненужные детали. Например, в диаграмму элемента можно добавить только те операции/атрибуты, которые важны для данной диаграммы, неважные особенности класса в UML можно опускать.

Для более полного раскрытия архитектуры проектируемой системы могут строиться диаграммы композитной/составной структуры и диаграммы автомата.

Диаграмма композитной/составной структуры

Диаграмма композитной/составной структуры — статическая структурная диаграмма, демонстрирует внутреннюю структуру классов и, по возможности, взаимодействие элементов (частей) внутренней структуры класса.

Подвидом диаграмм композитной структуры являются диаграммы кооперации (Collaboration diagram, введены в UML 2.0), которые показывают роли и взаимодействие классов в рамках кооперации. Кооперации удобны при моделировании шаблонов проектирования.

Диаграммы композитной структуры могут использоваться совместно с диаграммами классов, как показано на рисунке 16.

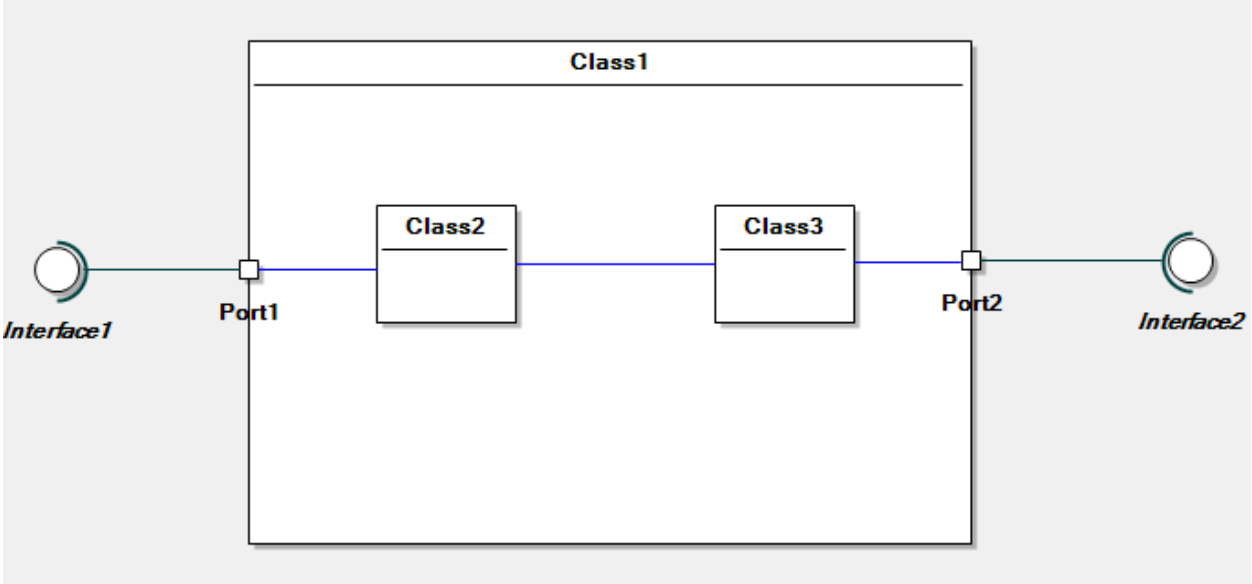


Рисунок 16 - Диаграмма композитной структуры, созданная в среде Borland Together

Диаграммы автомата

Диаграмма автомата, State Machine diagram (диаграмма конечного автомата, диаграмма состояний) — диаграмма, на которой представлен конечный автомат с простыми состояниями, переходами и композитными состояниями, как показано на рисунке 17.

Конечный автомат (State machine) — спецификация последовательности состояний, через которые проходит объект или взаимодействие в ответ на события своей жизни, а также ответные действия объекта на эти события. Конечный автомат прикреплён к исходному элементу (классу, кооперации или методу) и служит для определения поведения его экземпляров.

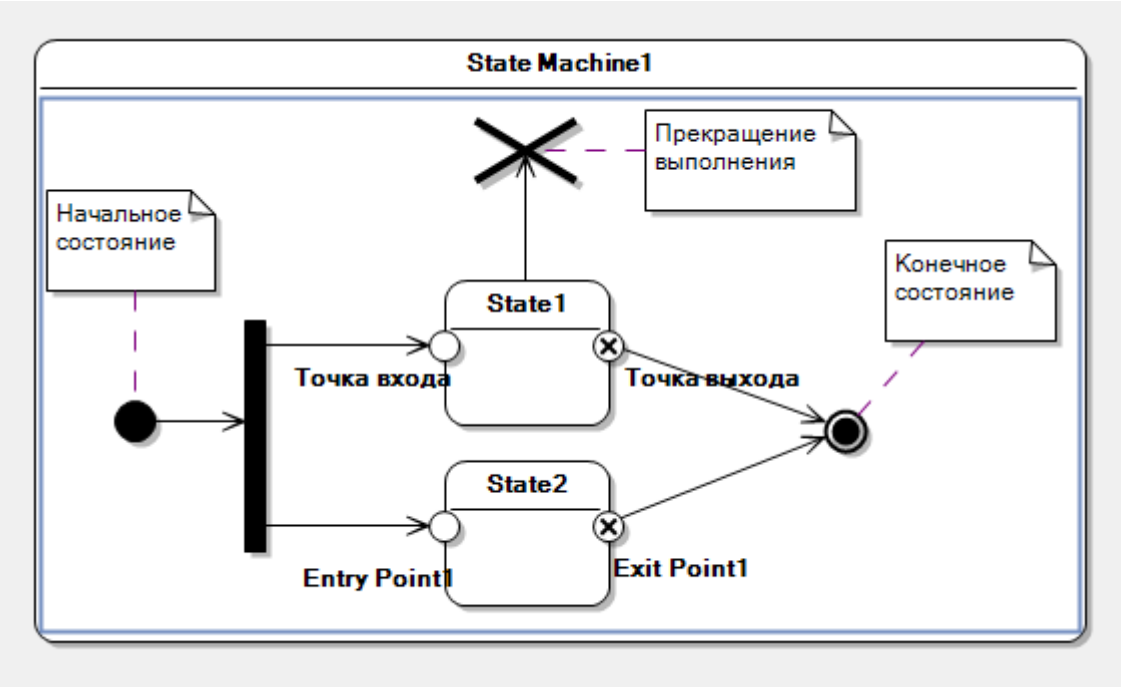


Рисунок 17 – Диаграмма автомата, созданная в среде Borland Together

Если в системе присутствуют объекты со сложным поведением, то строят диаграммы состояний. Построение диаграмм состояний может оказать следующее воздействие на описание классов:

-события могут отображаться в операции класса;

-особенности конкретных состояний могут повлиять на детали выполнения операций;

-описание состояний и переходов может помочь при определении атрибутов класса.

Каждая диаграмма состояний в UML описывает все возможные состояния одного экземпляра определенного класса и возможные последовательности его переходов из одного состояния в другое, то есть моделирует все изменения состояний объекта как его реакцию на внешние воздействия.

Диаграммы состояний чаще всего используются для описания поведения отдельных объектов, но также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

Главное предназначение этой диаграммы — описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

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

Действие рисуют вдоль линии перехода после имени события, его изображению предшествует наклонная (косая) черта.

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

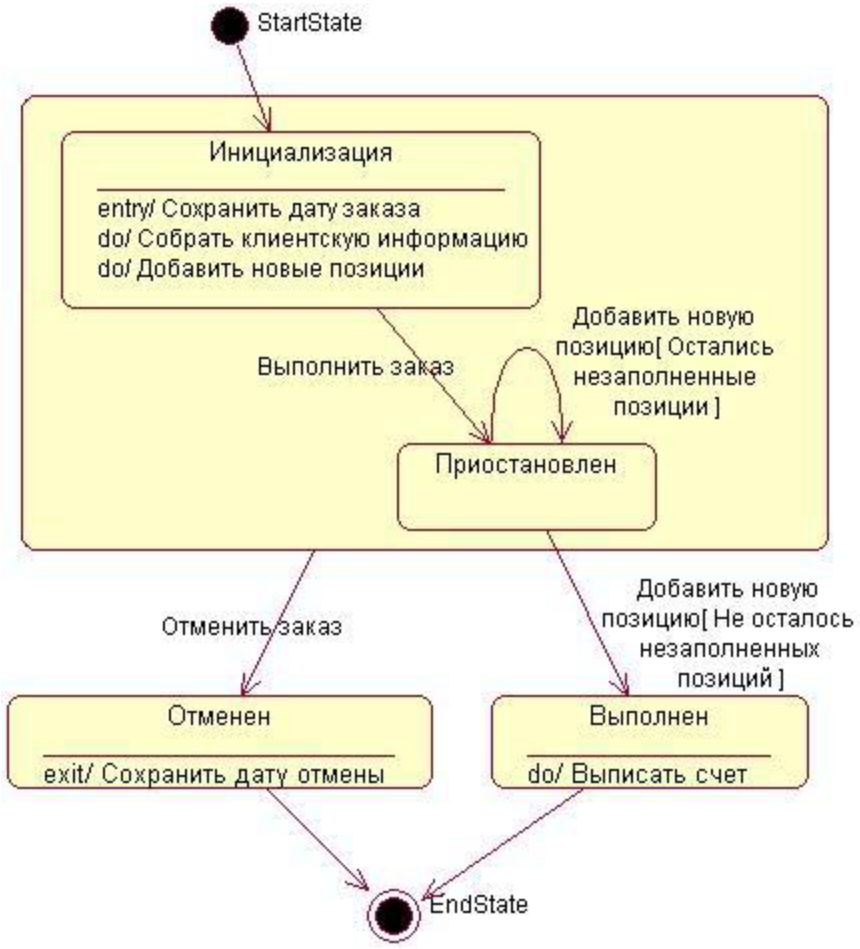


Рисунок 18 – Диаграмма состояний, созданная в среде Rational Rose

Для группировки классов, обладающих некоторой общностью, применяются пакеты. Пакет – общий механизм для организации элементов модели в группы. Каждый пакет – это группа элементов

модели, иногда сопровождаемая диаграммами, поясняющими структуру группы. Каждый элемент модели может входить только в один пакет. *Диаграммы пакетов* отображают зависимости между пакетами, возникающие, если элемент одного пакета зависит от элемента другого.

Пакеты также используются для представления подсистем. Подсистема – это комбинация пакета (поскольку она включает некоторое множество классов) и класса (поскольку она обладает поведением, т.е. реализует набор операций, которые определены в ее интерфейсах). Связь между подсистемой и интерфейсом называется связью реализации.

Жёсткого разделения между разными структурными диаграммами не проводится, поэтому данное название предлагается исключительно для удобства и не имеет семантического значения (пакеты и диаграммы пакетов могут присутствовать на других структурных диаграммах).

1.3.4 Реализация

Основная задача процесса реализации – создание системы в виде компонентов – исходных текстов программ, сценариев, двоичных файлов, исполняемых модулей и т.д. На этом этапе создается модель реализации, которая описывает то, как реализуются элементы модели проектирования, какие классы будут включены в конкретные компоненты. Данная модель описывает способ организации этих компонентов в соответствии с механизмами структурирования и разбиения на модули, принятыми в выбранной среде программирования и представляется диаграммой компонентов

Диаграмма компонентов

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Общий вид структуры информационной системы в виде диаграммы компонентов показан на рисунках 19 и 20. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ или реализации интерфейсов.

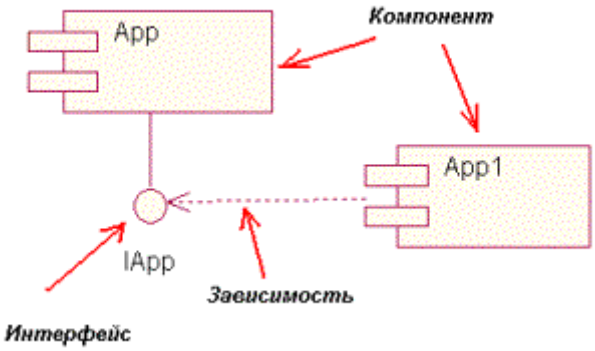


Рисунок 19 - Пример диаграммы компонентов на UML 1.5, созданной в среде Rational Rose

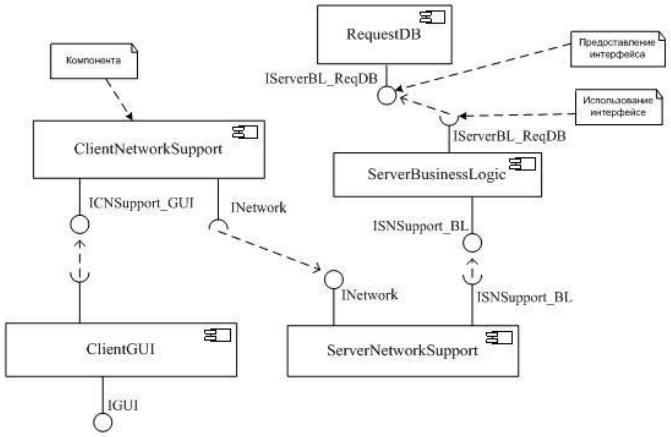


Рисунок 19 - Пример диаграммы компонентов на UML 2.0, созданной в среде Borland Together

Диаграмма развертывания

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Если разрабатывается программа, выполняющаяся локально на компьютере пользователя и не использующая периферийных устройств и ресурсов, то в разработке дополнительных диаграмм нет необходимости. При разработке же корпоративных приложений наличие таких диаграмм может быть крайне полезным для решения задач рационального размещения компонентов в целях эффективного использования распределенных вычислительных и коммуникационных ресурсов сети, обеспечения безопасности и других.

Для представления общей конфигурации и топологии распределенной программной системы в UML предназначены диаграммы развертывания.

Диаграмма развёртывания, Deployment diagram — служит для моделирования работающих узлов (аппаратных средств) и артефактов, развёрнутых на них. В UML 2 на узлах разворачиваются артефакты (artifact), в то время как в UML 1 на узлах разворачивались компоненты. Между артефактом и логическим элементом (компонентом), который он реализует, устанавливается зависимость манифестации. Это самый простой тип диаграмм, предназначенный для моделирования распределения устройств в сети. Для отображения используется всего два варианта значков процессор и устройство вместе со связями между ними.

Разработка диаграммы развертывания начинается с идентификации всех аппаратных, механических и других типов устройств, которые необходимы для выполнения системой всех своих функций. В первую очередь специфицируются вычислительные узлы системы, обладающие памятью и/или процессором.

Один из возможных вариантов построения диаграммы развертывания, созданной в среде Borland Together, показан на рисунке 21.

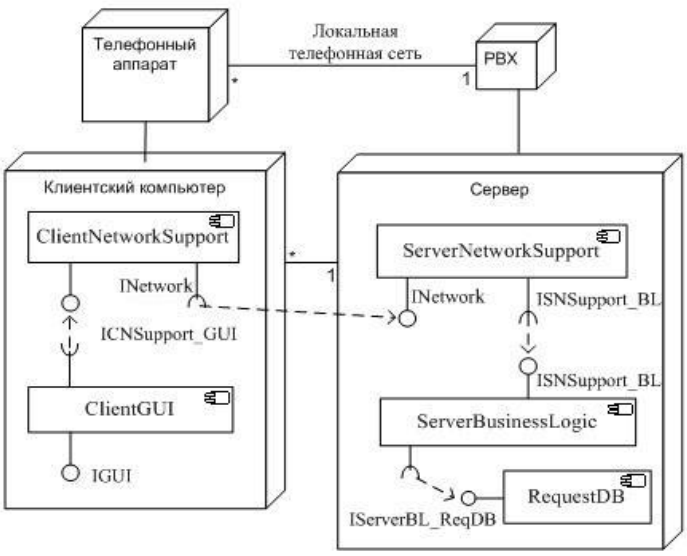


Рисунок 21 - Диаграмма развертывания, созданная в среде Borland Together

На диаграмме, показано каким образом компоненты телефонной службы приема заявок распределяются по аппаратной части системы.

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)				
6.1. Рекомендуемая литература				
6.1.1. Основная литература				
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л1.1	Игнатьев, С. А., Игнатьева, С. С.	Информационное обеспечение систем управления качеством: учебное пособие	Саратов: Саратовский государственный технический университет имени Ю.А. Гагарина, ЭБС АСВ, 2012	http://www.iprbookshop.ru/76484.html
Л1.2	Граничин О. Н., Кияев В. И.	Информационные технологии в управлении	Москва: Интернет-Университет Информационных Технологий	http://www.iprbookshop.ru/57379.html
Л1.3	Головицына М. В.	Информационные технологии в экономике	Москва: Интернет-Университет Информационных Технологий	http://www.iprbookshop.ru/52152.html
Л1.4	Малышева, Е. В.	Стратегическое планирование. Часть 1: учебное пособие	Новосибирск: Новосибирский государственный технический университет, 2010	http://www.iprbookshop.ru/45036.html
6.1.2. Дополнительная литература				
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л2.1	Гатина, Л. И.	Стратегическое планирование развития предприятии: учебно-методическое пособие	Казань: Казанский национальный исследовательский технологический университет, 2012	http://www.iprbookshop.ru/62291.html
Л2.2	Горбунов В. Л.	Бизнес-планирование	Москва: Интернет-Университет Информационных Технологий	http://www.iprbookshop.ru/56371.html
Л2.3	Силаенков, А. Н.	Информационное обеспечение и компьютерные технологии в научной и образовательной деятельности: учебное пособие	Омск: Омский государственный институт сервиса, Омский государственный технический университет, 2014	http://www.iprbookshop.ru/26682.html
6.1.3. Методические разработки				
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л3.1	Глазкова, И. Ю., Ловянников, Д. Г.	Информационные технологии в бизнес-планировании: лабораторный практикум	Ставрополь: Северо-Кавказский федеральный университет, 2017	http://www.iprbookshop.ru/75574.html
6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"				

Э1	Стасышин В.М. Проектирование информационных систем и баз данных [Электронный ресурс]: учебное пособие/ Стасышин В.М.— Электрон. текстовые данные.— Новосибирск: Новосибирский государственный технический университет, 2012.— 100 с.— Режим доступа: http://www.iprbookshop.ru/45001.— ЭБС «IPRbooks»
Э2	Тараненко Л.Г. Информационное обеспечение потребностей региона [Электронный ресурс]: учебное пособие/ Тараненко Л.Г.— Электрон. текстовые данные.— Кемерово: Кемеровский государственный институт культуры, 2009.— 194 с.— Режим доступа: http://www.iprbookshop.ru/21974.— ЭБС «IPRbooks»
Э3	Стратегическое планирование развития строительной организации [Электронный ресурс]/ А.Н. Асаул [и др.].— Электрон. текстовые данные.— СПб.: Институт проблем экономического возрождения, Санкт-Петербургский государственный архитектурно-строительный университет, 2009.— 166 с.— Режим доступа: http://www.iprbookshop.ru/18214.— ЭБС «IPRbooks»

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Технологический институт сервиса (филиал) ДГТУ в г.Ставрополе (ТИС (филиал) ДГТУ в г.Ставрополе)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Управление информационными ресурсами»
для студентов направления подготовки
09.04.02 Информационные системы и технологии
Направленность (профиль) Информационные системы и технологии

Методические указания по дисциплине «Управление информационными ресурсами» содержат задания для студентов, необходимые для практических занятий.

Проработка предложенных заданий позволит студентам приобрести необходимые знания в области изучаемой дисциплины.

Предназначены для студентов направления подготовки 09.04.02 Информационные системы и технологии (профиль) Информационные системы и технологии

Содержание

ВВЕДЕНИЕ

При изучении курса наряду с овладением студентами теоретическими положениями уделяется внимание приобретению практических навыков, с тем, чтобы они смогли успешно применять их в своей последующей работе.

Цель освоения дисциплины - формирование у обучаемых знаний в области теоретических основ информационной безопасности и навыков практического обеспечения защиты информации и безопасного использования программных средств в вычислительных системах, используемых на предприятиях.

В результате освоения данной дисциплины формируются следующие компетенции у обучающегося:

УК-2.1: Анализирует этапы жизненного цикла проекта, этапы разработки и реализации проекта

ПК-3.2: Осуществляет организационное и технологическое обеспечение проектирования информационных систем

ПК-2.2: Осуществляет анализ, синтез, оптимизацию и прогнозирование процессов функционирования информационных процессов

Изучив данный курс, студент должен:

Знать:

Теоретические основы управления информационными ресурсами, стандарты и подходы к управлению информационными ресурсами на предприятиях.

Уметь:

Работать со специальной литературой и нормативными документами (стандартами), работать со специальной литературой, использовать полученные знания для осуществления практической деятельности в области управления информационными ресурсами.

Владеть:

Современными подходами управления информационными ресурсами.

Реализация компетентностного подхода предусматривает широкое использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных ситуаций, собеседование) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков специалистов.

Лекционный курс является базой для последующего получения обучающимися практических навыков, которые приобретаются на практических занятиях, проводимых в активных формах: деловые игры; ситуационные семинары. Методика проведения практических занятий и их содержание продиктованы стремлением как можно эффективнее развивать у студентов мышление и интуицию, необходимые современному специалисту. Активные формы семинаров открывают большие возможности для проверки усвоения теоретического и практического материала.

Практическое занятие 1 Разработка элементарной сетевой модели управления информационными ресурсами

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание.1 Разработка элементарной сетевой модели управления информационными ресурсами

1. В настройках BIOS установите следующую последовательность загрузки устройств: CD-ROM Жесткий диск. Эта настройка всегда зависит от типа BIOS, поэтому ее нельзя описать универсально. Подробную информацию вы найдете в описании, прилагающийся к вашей материнской плате.
2. В привод CD-ROM вставьте установочный компакт-диск с операционной системой

Windows Server 2003 и перезагрузите компьютер.
3. Установка системы должна начаться автоматически. Если этого не происходит, проверьте еще раз порядок загрузки в BIOS. Если же в компьютере уже была установлена какая-то операционная система, может случиться так, что для начала установки системе будет требоваться нажатие любой клавиши.
4. Включится текстовый режим установки и появится окно с надписью Windows Server 2003 Setup (Установка операционной системы Windows).
5. Ознакомьтесь с информацией программы установки и нажмите Enter.
6. Ознакомьтесь с информацией программы установки и нажмите Enter.
7. Ознакомьтесь с лицензионным соглашением и согласитесь с ним (клавиша F8).
8. Создайте раздел для ОС на всем жестком диске клавишей ENTER.
9. Выполните форматирование созданного раздела в файловой системе NTFS - нажмите ENTER. Дождитесь окончания форматирования раздела, и копирования файлов установки на него. В процессе копирования компьютер перезагрузится и продолжит установку автоматически.
10. Самостоятельно укажите параметры языка и раскладки клавиатуры и перейдите к следующему шагу кнопкой «Далее».
11. Укажите регистрационные данные: ведите в поле Имя – USER о ведите в поле Организация – SIBCOL завершите ввод кнопкой «Далее».
12. Введите в поле Ключ продукта лицензионный ключ и щелкните «Далее».
13.Укажите вариант лицензирования, при котором для каждого подключения требуется отдельная лицензия: о установите радиокнопку на сервере; введите в текстовое поле количество одновременных подключений, например, 10; подтвердите параметры кнопкой «Далее».
14. Укажите имя компьютера и пароль администратора:

Введите в поле Имя компьютера – WIN2003;

Введите в поле Пароль администратора – 123456;

Введите в поле Подтверждение - 123456.

Подтвердите сделанные изменения кнопкой «Далее». Появится диалоговое окно, сообщающее о том, что пароль слишком простой. Ознакомьтесь с информацией о том, что вы указали простой пароль и продолжите установку кнопкой Да.
15. Укажите дату и время и щелкните «Далее».
16. Установите сетевые параметры для использования статического IPадреса: о выберите радиокнопку Обычные параметры и щелкните «Далее»;
17. Укажите сетевую группу, например, Workgroup и щелкните «Далее».
18. Дождитесь окончания выполнения установки ОС. По окончании установки компьютер перезагрузится. После этого загрузится операционная система Windows 2003 Server.

Задание.2 Формирование модели проекта. Формирование проектной группы. Система коммуникаций в процессе проектирования

Работа в рабочей группе

1. Щелкните правой кнопкой мыши на значке Мой компьютер, расположенном на Рабочем столе Windows, выберите в появившемся меню пункт Свойства
2. Перейдите ко вкладке Имя компьютера
3. Щелкните мышью на кнопке Изменить
4. Компьютер входит в сетевую рабочую группу, выберите режим Рабочей группы и наберите ее название в расположенном рядом поле.
5. Создать папку и ограничьте доступ следующим образом:

ПК 1 имеет доступ к ПК 3,4,6 на чтение и запись, к ПК7 на чтение, к остальным доступа не имеет.

ПК 2 имеет доступ к ПК 5,8 на чтение и запись, к ПК5 на чтение, к остальным доступа не имеет.

ПК 3 имеет доступ к ПК 7,9 на чтение и запись, к ПК4 на чтение, к остальным доступа не

имеет.
ПК 4 имеет доступ к ПК 1,2 на чтение и запись, к ПК3 на чтение, к остальным доступа не имеет.
ПК 5 имеет доступ к ПК 4,7 на чтение и запись, к ПК2 на чтение, к остальным доступа не имеет.
ПК 6 имеет доступ к ПК 5,9 на чтение и запись, к ПК6 на чтение, к остальным доступа не имеет.
ПК 7 имеет доступ к ПК 6,8 на чтение и запись, к ПК8 на чтение, к остальным доступа не имеет.
ПК 8 имеет доступ к ПК 7,3 на чтение и запись, к ПК9 на чтение, к остальным доступа не имеет.
ПК 9 имеет доступ к ПК 2,6 на чтение и запись, к ПК10 на чтение, к остальным доступа не имеет.
ПК 10 имеет доступ к ПК 4,6 на чтение и запись, к ПК1 на чтение, к остальным доступа не имеет.
6.Заблокировать настройки рабочего стола.
7.Заблокировать сетевые настройки.
8.Создать папку на рабочем столе и сделать к ней общий доступ для всех на чтение.

Контрольные вопросы

1. Охарактеризуйте место операционной системы в программном обеспечении компьютеров, компьютерных систем и сетей.
2. В чем заключается основное назначение операционной системы?
3. Перечислите основные функции операционной системы.
4. Дайте понятие компьютерных ресурсов.
5. Дайте определение архитектуры операционных систем.
6. Перечислите поколения операционных систем.
7. Перечислите классификационные признаки операционной системы.
8. Охарактеризуйте виды интерфейсов операционных систем.
9. Опишите особенности эволюционных этапов операционных систем.
10. В чем заключается эффективность операционной системы?

Практическое занятие 2 Формирование модели проекта. Формирование проектной группы. Система коммуникаций в процессе проектирования

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание.

2.1. Ознакомьтесь с теоретическими основами защиты информации в ОС семейства Windows в настоящих указаниях и конспектах лекций.

2.2. Выполните задания 2.2.1-2.2.8 2.2.1.

2.2.1 При выполнении практического задания запустите в программе Oracle VM Virtualbox виртуальную машину Win7Test. Войдите в систему под учетной записью администратора. Все действия в пп 2.2.1-2.2.8 выполняйте в системе, работающей на виртуальной машине.

2.2.2. Создайте учетную запись нового пользователя testUser в оснастке «Управление компьютером» (compmgmt.msc). При создании новой учетной 22 записи запретите пользователю смену пароля и снимите ограничение на срок действия его пароля. Создайте новую группу ”testGroup” и включите в нее нового пользователя. Удалите пользователя из других групп. Создайте на диске C: папку forTesting. Создайте или скопируйте в эту папку несколько текстовых файлов (*.txt).

2.2.3. С помощью команды runas запустите сеанс командной строки (cmd.exe) от имени вновь созданного пользователя. Командой whoami посмотрите SID пользователя и всех его групп, а также текущие привилегии пользователя. Строку запуска и результат работы этой и всех следующих консольных команд копируйте в файл протокола лабораторной работы.

2.2.4. Убедитесь в соответствии имени пользователя и полученного SID в реестре Windows. Найдите в реестре, какому пользователю в системе присвоен SID S-1-5-21-1957994488-492894223-170857768-1004 (Используйте ключ реестра HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList).

2.2.5. Командой whoami определите перечень текущих привилегий пользователя testUser. В сеансе командной строки пользователя попробуйте изменить системное время командой time. Чтобы предоставить пользователю подобную привилегию, запустите оснастку «Локальные параметры безопасности» (secpol.msc). Добавьте пользователя в список параметров политики «Изменение системного времени» раздела Локальные политики -> Назначение прав пользователя. После этого перезапустите сеанс командной строки от имени пользователя, убедитесь, что в списке привилегий добавилась SeSystemtimePriviege. Попробуйте изменить системное время командой time. Убедитесь, что привилегия «Завершение работы системы» (SeShutdownPrivilege) предоставлена пользователю testUser . После этого попробуйте завершить работу системы из сеанса командной строки пользователя командой shutdown –s. Добавть ему привелегию «Принудительное удаленное завершение» (SeRemoteShutdownPrivilege). Попробуйте завершить работу консольной командой еще раз (отменить команду завершения до ее непосредственного выполнения можно командой shutdown –a).

2.2.6. Ознакомьтесь с справкой по консольной команде icacls. Используя эту команду, просмотрите разрешения на папку c:\forTesting. Объясните все обозначения в описаниях прав пользователей и групп в выдаче команды. а) Разрешите пользователю testUser запись в папку forTesting, но запретите запись для группы testGroup. Попробуйте записать файлы или папки в forTesting от имени пользователя testUser. Объясните результат. Посмотрите эффективные разрешения пользователя testUser к папке forTesting в окне свойств папки. б) Используя стандартное окно свойств папки, задайте для пользователя testUser такие права доступа к папке, чтобы он мог записывать информацию в папку forTesting, но не мог просматривать ее содержимое. Проверьте, что папка forTesting является теперь для пользователя testUser “слепой”, запустив, например, от его имени файловый менеджер и попробовав записать файлы в папку, просмотреть ее содержимое, удалить файл из папки. в) Для вложенной папки forTesting\Docs отмените наследование ACL от родителя и разрешите пользователю просмотр, чтение и запись в папку. Проверьте, что для пользователя папка forTesting\Docs перестала быть “слепой” (например, 23 сделайте ее текущей в сеансе работы файлового менеджера от имени пользователя и создайте в ней новый файл). г) Снимите запрет на чтение папки forTesting

для пользователя testUser. Используя команду icacls запретите этому пользователю доступ к файлам с расширением txt в папке forTesting. Убедитесь в недоступности файлов для пользователя. д) Командой icacls запретите пользователю все права на доступ к папке forTesting и разреште полный доступ к вложенной папке forTesting\Docs. Убедитесь в доступности папки forTesting\Docs для пользователя. Удалите у пользователя testUser привилегию SeChangeNotifyPrivilege. Попробуйте получить доступ к папке forTesting\Docs. Объясните результат. е) Запустите файловый менеджер от имени пользователя testUser и создайте в нем папку newFolder на диске C. Для папки newFolder очистите весь список ACL командой cacls. Попробуйте теперь получить доступ к папке от имени администратора и от имени пользователя. Кто и как теперь может вернуть доступ к папке? Верните полный доступ к папке для всех пользователей. ж) Создайте в разделе HKLM\Software реестра раздел testKey. Запретите пользователю testUser создание новых разделов в этом разделе реестра. Создайте для раздела HKLM\Software\testKey SACL, позволяющий протоколировать отказы при создании новых подразделов, а также успехи при перечислении подразделов и запросе значений (предварительно проверьте, что в локальной политике безопасности соответствующий тип аудита включен). Попробуйте от имени пользователя testUser запустить regedit.exe и создать раздел в HKLM\Software. Убедитесь, что записи аудита были размещены в журнале безопасности (eventvwr.msc). з) С использованием команды whoami проверьте уровень целостности для пользователя testUser и администратора (учетная запись ВПИ). Запустите какоенибудь приложение (калькулятор, блокнот) от имени testUser и администратора. С использованием утилиты ProcessExplorer (можно найти в папке c:\Utils на виртуальной машине) проверьте уровень целостности запущенных приложений. Объясните разницу. Верните пользователю testUser права на полный доступ к папке forTesting. От имени администратора создайте в папке forTesting текстовый файл someText.txt. Измените уровень целостности этого файла до высокого с использованием команды icacls. Запустите блокнот от имени пользователя testUser, откройте в нём файл someText.txt, измените содержимое файла и попробуйте сохранить изменения. Объясните причину отказа в доступе. Как можно предоставить пользователю testUset доступ к файлу.

2.2.7. Шифрование файлов и папок средствами EFS. а) От имени пользователя testUser зашифруйте какой-нибудь файл на диске. Убедитесь, что после этого был создан сертификат пользователя, запустив оснастку certmgr.msc от имени пользователя (раздел Личные). Просмотрите основные параметры сертификата открытого ключа пользователя testUser (срок действия, используемые алгоритмы). Установите доверие к этому сертификату в вашей системе. б) Создайте в папке forTesting новую папку Encrypt. В папке Encrypt создайте или скопируйте в нее текстовый файл. Зашифруйте папку Encrypt и все ее содержимое из меню свойств папки от имени администратора. Попробуйте про24 смотреть или скопировать какой-нибудь файл этой папки от имени пользователя testUser. Объясните результат. Скопируйте зашифрованный файл в незашифрованную папку (например, forTesting). Убедитесь, что он остался зашифрованным. Добавьте пользователя testUser в список имеющих доступа к файлу пользователей в окне свойств шифрования файла. Повторите попытку получить доступ к файлу от имени пользователя testUser. в) Создайте учетную запись нового пользователя agentUser, сделайте его членом группы Администраторы. Определите для пользователя agentUser роль агента восстановления EFS. Создайте в папке forTesting новый текстовый файл с произвольным содержимым. Зашифруйте этот файл от имени пользователя testUser. Убедитесь в окне подробностей шифрования файла, что пользователь agentUser является агентом восстановления для данного файла. Попробуйте прочитать содержимое файла от имени администратора и от имени пользователя agentUser. Объясните результат. г) Зашифруйте все текстовые файлы папки forTesting с использованием консольной команды шифрования cipher от имени пользователя testUser (предварительно снимите запрет на доступ к этим файлам, установленный в задании 2.2.6г). д) Убедитесь, что при копировании зашифрованных

файлов на том с файловой системой, не поддерживающей EFS (например, FAT32 на флеш-накопителе), содержимое файла дешифруется.

2.2.8. После демонстрации результатов работы преподавателю восстановите исходное состояние системы: удалите созданные папки и файлы, разделы реестра, удалите учетную запись созданного пользователя и его группы, снимите с пользователя agentUser роль агента восстановления.

Контрольные вопросы

1. К какому классу безопасности относится ОС Windows по различным критериям оценки.
2. Каким образом пользователи идентифицируются в ОС Windows.
3. Что такое списки DACL и SACL.
4. Перечислите, каким образом можно запустить процесс от имени другого пользователя.
5. Как происходит проверка прав доступа пользователя к ресурсам в ОС Windows.
6. Что такое маркер безопасности, и какова его роль в модели безопасности Windows.
7. Как с использованием команды icacls добавить права на запись для всех файлов заданной папки.
8. Что такое уровень целостности? Как он влияет на права доступа субъ-ектов к объектам ОС? Как можно узнать и задать уровень целостности для объек-тов и субъектов?
9. Какие события подлежат аудиту в ОС Windows.
10. Каким образом шифруются файлы в файловой системе EFS? Что такое FEK? DDF? DDR.
11. Какие алгоритмы шифрования используются в EFS.

Практическое занятие 3 Оценка качества ресурсов методом экспертных оценок

Цель занятия заключается в формировании у студентов профессиональной компетенции: ПК-4.1

Задание1. Настройка и просмотр сведений о системе

Чтобы запустить программу «Сведения о системе», нажмите кнопку Пуск и выберите команду Справка и поддержка. Нажмите кнопку Поддержка на панели инструментов, затем щелкните ссылку Расширенные сведения о системе в группе Средства и ссылки в левой части окна. В правой части окна щелкните ссылку Просмотр дополнительных сведений о системе.

Настройка системы.

Чтобы запустить программу «MSconfig.exe», нажмите кнопку Пуск и выберите команду Справка и поддержка. Нажмите кнопку Поддержка на панели инструментов, затем щелкните ссылку Настройка системы в группе Средства и ссылки в левой части окна. В правой части окна щелкните ссылку Запуск программы настройки системы

После загрузки появляется окно с шестью вкладками:

- Общие - позволяет управлять параметрами запуска системы.
- Config.sys - редактирование файла config.sys.
- Autoexec.bat - соответственно.
- System.ini.
- Win.ini.

Задание2. Автозагрузка файлов

Автозагрузка - здесь перечислены все программы, которые запускаются при загрузке системы.

Очень удобно то, что все собрано в одном месте. Не надо лазить по реестру и файлам, чтобы посмотреть, что загружается на компьютере. Можно отключить загрузку

любой программы или выполнение строки одного из перечисленных файлов, не правя ничего вручную. При этом комментарии будут расставлены автоматически, а программы, запускаемые из реестра, например, из раздела "Run", будут перенесены в раздел "Run-" (в конце соответствующего раздела добавляется символ "-").

Специальный текстовый конфигурационный файл «BOOT.INI», который используется в процессе загрузки — один из важнейших системных файлов «Windows XP».

Этот файл должен находиться в корневом каталоге загрузочного диска. Перед тем как модифицировать файл измените его атрибуты, так чтобы он не был «Только для чтения» (щёлкните правой кнопкой мыши по файлу и выберите в контекстном меню последний пункт — «Свойства» и скиньте соответствующий флажок, устанавливаемый по умолчанию при инсталляции ОС).

Раздел [boot loader] служит для задания параметров загрузки операционной системы.

Параметр «timeout = 30» (по умолчанию) определяет количество секунд, в течение которого пользователь может выбирать один из пунктов меню. При «timeout = 0» загрузочное меню не отображается. «При timeout = -1» меню находится на экране неограниченное время.

Параметр «default =» определяет путь к загружаемой по умолчанию системе. В разделе [operation systems] находятся сведения об установленных операционных системах.

При использовании двух операционных систем, например, «Windows Me» и «Windows XP», содержимое файла будет выглядеть примерно так:

[boot loader]

timeout=30

default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS

[operating systems]

multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional RU" /noexecute=optin /fastdetect"

Здесь:

«multi(0)» — порядковый номер адаптера, с которого осуществляется загрузка. Всегда имеет значение «0»,

«disk(0)» — всегда равен «0» (для большинства BIOS),

«rdisk(X)» — определяет порядковый номер жесткого диска с которого осуществляется загрузка (от «0» до «3»),

«partition(Y)» — порядковый номер раздела жесткого диска, с которого загружается ОС. Нумерация начинается с «1». Не нумеруются расширенные разделы MS-DOS (тип «5») и разделы типа «0 » — неиспользуемые.

Способы автозагрузки и отключение списков автозагрузки:

Реестр - в реестре автозагрузка представлена в нескольких местах:

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRun] - программы, которые запускаются при входе в систему. Данный раздел отвечает за запуск программ для всех пользователей системы.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunOnce] - программы, которые запускаются только один раз при входе пользователя в систему. После этого ключи программ автоматически удаляются из данного раздела реестра. Данный раздел отвечает за запуск программ для всех пользователей системы.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunOnceEx] - программы, которые запускаются только один раз, когда загружается система. Этот раздел используется при инсталляции программ, например для запуска настроечных модулей.

После этого ключи программ автоматически удаляются из данного раздела реестра. Данный раздел отвечает за запуск программ для всех пользователей системы.

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRun]- программы, которые запускаются при входе текущего пользователя в систему

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRunOnce] - программы, которые запускаются только один раз при входе текущего пользователя в систему. После этого ключи программ автоматически удаляются из данного раздела реестра.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunServices] - программы, которые загружаются при старте системы до входа пользователя в Windows.

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionRunServicesOnce] - программы отсюда загружаются только один раз, когда загружается система.

Например, чтобы автоматически запускать Блокнот при входе текущего пользователя, открываем Редактор реестра (regedit.exe), переходим в раздел

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionRun]

и добавляем следующий ключ:

"NOTEPAD.EXE"="C:WINDOWSSystem32notepad.exe"

Откройте оснастку "Групповая политика" (gpedit.msc), перейдите на вкладку "Конфигурация компьютера - Административные шаблоны - Система". В правой части оснастки перейдите на пункт "Запускать указанные программы при входе в систему". По умолчанию эта политика не задана, но вы можете добавить туда программу: включаем политику, нажимаем кнопку "Показать - Добавить", указываем путь к программе, при этом если запускаемая программа находится в папке. WINDOWSSystem32 то можно указать только название программы, иначе придется указать полный путь к программе. При этом в системном реестре в разделе [HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionpolicies] создается подраздел ExplorerRun с ключами добавленных программ. Пример:

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionpoliciesExplorerRun]

"1"="notepad.exe"

"2"="iexplore.exe"

В итоге получаем запуск Блокнота и Internet Explorer для всех пользователей.

Аналогично задается автозапуск для текущих пользователей, в оснастке "Групповая политика" это путь "Конфигурация пользователя - Административные шаблоны - Система", а в реестре раздел

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionPoliciesExplorerRun]

При этом программы из этого списка не отображаются в списке программ, доступных для отключения в msconfig.exe, а также определяются не всеми менеджерами автозагрузки.

6. Папка "Автозагрузка"- это папка, в которой хранятся ярлыки для программ запускаемых после входа пользователя в систему. Ярлыки в эту папку могут добавляться программами при их установке или пользователем самостоятельно. Существует две папки - общая для всех пользователей и индивидуальная для текущего пользователя. По умолчанию эти папки находятся здесь:

.Documents and SettingsAll UsersГлавное менюПрограммы Автозагрузка - это папка, программы из которой будут запускаться для всех пользователей компьютера.

.Documents and SettingsUsernameГлавное менюПрограммыАвтозагрузка- это папка, программы из которой будут запускаться для текущего пользователя (здесь он назван Username).

Посмотреть, какие программы у вас запускаются таким способом, можно, открыв меню "Пуск - Все программы - Автозагрузка". Если вы создадите в этой папке ярлык для какой-нибудь программы, она будет запускаться автоматически после входа пользователя в систему. Если при входе пользователя в систему удерживать нажатой клавишу "Shift",

то программы из папок "Автозагрузка" запускаться не будут.

7. Смена папки автозагрузки- Windows считывает данные о пути к папке "Автозагрузка" из реестра. Этот путь прописан в следующих разделах:

[HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionExplorerUser Shell Folders]

"Common Startup"="%ALLUSERSPROFILE%Главное менюПрограммыАвтозагрузка" - для всех пользователей системы.

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionExplorerUser Shell Folders]

"Startup"="%USERPROFILE%Главное менюПрограммыАвтозагрузка" - для текущего пользователя.

Сменив путь к папке, мы получим автозагрузку всех программ из указанной папки. Например:

[HKEY_CURRENT_USERSoftwareMicrosoftWindowsCurrentVersionExplorerUser Shell Folders]

"Startup"="c:mystartup" - система загрузит все программы, ярлыки которых находятся в папке c:mystartup, при этом папка "Автозагрузка" все так же будет отображаться в меню "Пуск", а если у пользователя в ней ничего не было, то он и не заметит подмены.

СПИСОК РЕКОМЕНДУЕМЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

6.1. Рекомендуемая литература

6.1.1. Основная литература

	Авторы, составители	Заглавие	Издательство, год	Адрес
Л1.1	Селетков С. Н., Днепровская Н. В., Тультаева И. В.	Мировые информационные ресурсы и ресурсы знаний: учебно-практическое пособие: учебное пособие	Москва: Евразийский открытый институт, 2009	https://biblioclub.ru/index.php?page=book&id=90403
Л1.2	Днепровская Н. В., Селетков С. Н.	Мировые информационные ресурсы: учебно-методический комплекс	Москва: Евразийский открытый институт, 2010	http://biblioclub.ru/index.php?page=book&id=90406
Л1.3	Порядина О. В.	Управление информационными ресурсами: учебно-методическое пособие	Йошкар-Ола: Поволжский государственный технологический университет, 2015	https://biblioclub.ru/index.php?page=book&id=439328

6.1.2. Дополнительная литература

	Авторы, составители	Заглавие	Издательство, год	Адрес
	Авторы, составители	Заглавие	Издательство, год	Адрес
Л2.1	Яковенко, Л. В.	Управление информационными ресурсами: методическое пособие для бакалавров по специальности 6.030502 «экономическая кибернетика»	Симферополь: Университет экономики и управления, 2012	http://www.iprbookshop.ru/54718.html
Л2.2		Отраслевые информационные ресурсы: учебно-методический комплекс	Кемерово: Кемеровский государственный университет культуры и искусств (КемГУКИ), 2014	https://biblioclub.ru/index.php?page=book&id=279469

6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"

Э1	Яковенко Л.В. Управление информационными ресурсами [Электронный ресурс]: методическое пособие для бакалавров по специальности 6.030502 «Экономическая кибернетика»/ Яковенко Л.В.— Электрон. текстовые данные.— Симферополь: Университет экономики и управления, 2012.— 118 с.
Э2	Моделирование информационных ресурсов [Электронный ресурс]: учебно-методический комплекс по специальности 080801 «Прикладная информатика (в информационной сфере)», специализации «Информационные сети и системы», квалификация – «информатик-аналитик»/ — Электрон. текстовые данные.— Кемерово: Кемеровский государственный институт культуры, 2013.— 36 с.
Э3	Бирюков А.Н. Процессы управления информационными технологиями [Электронный ресурс]/ Бирюков А.Н.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 263 с

6.3.1 Перечень программного обеспечения

6.3.1.1	программное обеспечение: Windows 7 корпоративная, Microsoft Office 2007, Visual Studio 2013, MATLAB R2009b
6.3.1.2	зал электронной библиотеки ТИС

6.3.2 Перечень информационных справочных систем

6.3.2.1	1 Электронно-библиотечная система IPRbooks www.iprbookshop.ru
6.3.2.2	2 Универсальная библиотека онлайн www.BiblioClub.ru
6.3.2.3	3 Электронная библиотечная система www.znanium.com
6.3.2.4	4 Национальный цифровой ресурс www.rucont.ru