МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«БРАТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Базовая кафедра менеджмента и информационных технологий

УТВЕРЖДАЮ:
Проректор по учебной работе
_____Е.И. Луковникова
«_____» декабря 2018 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

РАЗРАБОТКА ПРОГРАММНЫХ ПРИЛОЖЕНИЙ

Б1.В.09

НАПРАВЛЕНИЕ ПОДГОТОВКИ

09.03.03 Прикладная информатика

ПРОФИЛЬ ПОДГОТОВКИ

Прикладная информатика в экономике

Программа академического бакалавриата

Квалификация (степень) выпускника: бакалавр

СОДЕРЖАНИЕ ПРОГРАММЫ **Стр.**

1. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Вид деятельности выпускника

Дисциплина охватывает круг вопросов, относящихся к проектному виду профессиональной деятельности выпускника в соответствии с компетенциями и видами деятельности, указанными в учебном плане.

Цель дисциплины

Формирование у обучающихся практических навыков по разработке программного обеспечения (ПО) для решения экономических и расчетных задач с применением современных методов и технологий программирования.

Основными задачами дисциплины являются:

− знакомство с основными концепциями разработки приложений информационных систем и получение практических навыков в работе с ними;

− получение практических навыков проектирования и разработки программных приложений.

Код компетенции	Содержание компетенций	Перечень планируемых результатов обучения по дисциплине
1	2	3
ПК-2	способность разрабатывать, внедрять и адаптировать прикладное программное обеспечение	знать: − состав функциональных и обеспечивающих подсистем ИС; − основные подходы и методы проектирования и создания ИС − основные среды для разработки программного обеспечения **уметь:** − проводить формализацию и реализацию решения прикладных задач; − моделировать схемы баз данных; − внедрять и адаптировать прикладное программное обеспечение; владеть: − современными языками программирования, методиками разработки и внедрения прикладного программного обеспечения;

ПК-7	способность проводить описание прикладных процессов и информационного обеспечения решения прикладных задач	**знать:** − методы анализа и моделирования бизнес-процессов − основные стандарты, технологии и нотации моделирования бизнес-процессов − инструментальные системы, используемые для описания и анализа бизнес-процессов; **уметь:** − моделировать, анализировать и совершенствовать бизнес-процессы с использованием изученных стандартов, технологий и нотаций моделирования; − рецензировать модель бизнес-процесса; **владеть:** − практическими навыками моделирования, анализа и документирования бизнес-процессов с помощью инструментальных сред; − терминологией из области моделирования бизнес-процессов; − методами построения, анализа и документирования моделей бизнес-процессов − навыками работы с инструментальными средствами моделирования предметной области, прикладных и информационных процессов
ПК-8	способность программировать приложения и создавать программные прототипы решения прикладных задач	**знать:** − модели и процессы жизненного цикла; − стадии создания ИС; **уметь:** − выполнять работы на всех стадиях жизненного цикла ИС; − разрабатывать концептуальную модель предметной области − выполнять работы по созданию технической документации; **владеть:** − современными языками программирования, методами разработки программных прототипов решения прикладных задач.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина Б1.В.09 Разработка программных приложений относится к вариативной.

Дисциплина Разработка программных приложений базируется на знаниях, полученных при изучении дисциплин: Информатика и программирование, Базы данных, Информационные системы и технологии, Интернет-программирование, Информационные системы в экономике, Высокоуровневые методы информатики и программирования.

Основываясь на изучении перечисленных дисциплин, Разработка программных приложений представляет основу для изучения дисциплин: Системная архитектура информационных систем; Проектный практикум; Программная инженерия.

Такое системное междисциплинарное изучение направлено на достижение требуемого ФГОС уровня подготовки по квалификации бакалавр.

3. РАСПРЕДЕЛЕНИЕ ОБЪЕМА ДИСЦИПЛИНЫ

3.1. Распределение объема дисциплины по формам обучения

Форма обучения	*Курс*	*Семестр*	*Трудоемкость дисциплины в часах*						*Курсовая работа*	*Вид промежуточной аттестации*
			Всего часов (экз)	*Аудиторных часов*	*Лекции*	*Лабораторные работы*	*Практические занятия*	*Самостоятельная работа*		
1	2	3	4	5	6	7	8	9	10	11
Очная	3	6	144	36	18	18	-	72	КР	экзамен
Заочная	3	-	144	16	4	-	12	119	КР	экзамен
Заочная (ускоренное обучение)	-	-	-	-	-	-	-	-	-	-
Очно-заочная	-	-	-	-	-	-	-	-	-	-

3.2. Распределение объема дисциплины по видам учебных занятий и трудоемкости

Вид учебных занятий	*Трудо-емкость (час.)*	*в т.ч. в интерактивной, активной, иннова-ционной формах, (час.)*	*Распределение по семестрам, час*
			6
1	2	3	4
. Контактная работа обучающихся с преподавателем (всего)	36	14	36
Лекции (Лк)	18	6	18
Лабораторные работы (ЛР)	18	8	18
Курсовая работа	+	-	+

Групповые консультации	+	-	+
I.Самостоятельная работа обучающихся (СР)	72	-	72
Подготовка к лабораторным занятиям	24	-	24
Выполнение курсовой работы	24	-	24
Подготовка к экзамену в течение семестра	24	-	24-
III. Промежуточная аттестация			
экзамен	36	-	36
Общая трудоемкость дисциплины . час.	144	-	144
зач. ед.	4	-	4

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Распределение разделов дисциплины по видам учебных занятий

- для очной формы обучения:

№ раз-дела и темы	*Наименование раздела*	*Трудоем-кость, (час.)*	*Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)*		
			учебные занятия		*самостоятельная работа обучаю-щихся*
			лекции	*Лабораторные работы*	
1	2	3	4	5	6
1.	**Основы разработки программных приложений**	**24**	**4**	**4**	**16**
1.1	Информационные технологии. Жизненный цикл информационной системы	3	1	-	2
1.2	Модели жизненного цикла	7	1	1	5
1.3	Основные принципы проектирования	10	1	3	6
1.4	Классификация моделей информационной системы	4	1	-	3
2.	**Технологии и подходы к проектированию программных приложений**	**19**	**3**	**2**	**14**
2.1	CASE-технологии анализа и проектирования	4	1	-	3
2.2	Сущность структурного анализа и проектирования	6	1	-	5
2.3	Разработка функциональной модели	9	1	2	6
3.	**Разработка информационной модели**	**25**	**3**	**4**	**18**
3.1	Основы проектирования баз данных	3,5	0,5	-	3
3.2	Концептуальное проектирование с использованием методологии IDEF1X	9	1	2	6
3.3	Логическое проектирование с использованием методологии IDEF1X	7	1	1	5

3.4	Физическое проектирование с использованием методологии IDEF1X	5,5	0,5	1	4
4.	**Основы объектно-ориентированного подхода к проектированию и разработке программ**	**40**	**8**	**8**	**24**
4.1	Объектно-ориентированное проектирование программ	1,5	0,5	-	1
4.2	Основные принципы объектно-ориентированного программирования. Классы и объекты, поля, свойства, методы, события. Конструкторы и деструкторы.	3	2	-	1
4.3	Проект, файлы, входящие в состав проекта.	2,5	0,5	1	1
4.4	Форма: свойства и методы формы, события, организация, реакция на них	4,5	0,5	1	3
4.5	Визуальные компоненты, использование библиотеки VCL/	5	1	1	3
4.6	Событие, обработчик события, создание и использование	5	1	1	3
4.7	Разработка графического интерфейса. Развитые элементы интерфейса.	4,5	0,5	1	3
4.8	Компоненты для ввода, отображения, редактирования и вывода информации	5	1	1	3
4.9	Элементы управления на форме. Работа с меню: главное контекстное, системное	4,5	0,5	1	3
4.10	Файлы, окна диалога работы с файлами. Настройка окон диалога.	4,5	0,5	1	3
	ИТОГО	**108**	**18**	**18**	**72**

- для заочной формы обучения:

№ раз-дела и темы	***Наименование раздела***	***Трудоем-кость, (час.)***	***Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)***		
			учебные занятия		***самостоятельная работа обучающихся***
			лекции	***Практические занятия***	
1	**2**	**3**	**4**	**5**	**6**
1.	**Основы разработки программных приложений**	**27**	**1**	**2**	**24**
1.1	Информационные технологии. Жизненный цикл информационной системы	5,25	0,25	-	5
1.2	Модели жизненного цикла	6,25	0,25	1	5
1.3	Основные принципы проектирования	11,25	0,25	1	10
1.4	Классификация моделей информационной системы	4,25	0,25	-	4

2.	**Технологии и подходы к проектированию программных приложений**	**27**	**1**	**2**	**24**
2.1	CASE-технологии анализа и проектирования	3,35	0,35	-	3
2.2	Сущность структурного анализа и проектирования	5,3	0,30	-	5
2.3	Разработка функциональной модели	18,35	0,35	2	16
3.	**Разработка информационной модели**	**36**	**1**	**4**	**31**
3.1	Основы проектирования баз данных	3,5	0,5	-	3
3.2	Концептуальное проектирование с использованием методологии IDEF1X	9	1	2	6
3.3	Логическое проектирование с использованием методологии IDEF1X	7	1	1	5
3.4	Физическое проектирование с использованием методологии IDEF1X	5,5	0,5	1	4
4.	**Основы объектно-ориентированного подхода к проектированию и разработке программ**	**45**	**1**	**4**	**40**
4.11	Объектно-ориентированное проектирование программ	4,1	0,1	-	4
4.12	Основные принципы объектно-ориентированного программирования. Классы и объекты, поля, свойства, методы, события. Конструкторы и деструкторы.	4,1	0,1	-	4
4.13	Проект, файлы, входящие в состав проекта.	4,6	0,1	0,5	4
4.14	Форма: свойства и методы формы, события, организация, реакция на них	4,6	0,1	0,5	4
4.15	Визуальные компоненты, использование библиотеки VCL/	4,6	0,1	0,5	4
4.16	Событие, обработчик события, создание и использование	6,6	0,1	0,5	6
4.17	Разработка графического интерфейса. Развитые элементы интерфейса.	4,6	0,1	0,5	4
4.18	Компоненты для ввода, отображения, редактирования и вывода информации	4,6	0,1	0,5	4
4.19	Элементы управления на форме. Работа с меню: главное контекстное, системное	3,6	0,1	0,5	3
4.20	Файлы, окна диалога работы с файлами. Настройка окон диалога.	3,6	0,1	0,5	3
	ИТОГО	**135**	**4**	**12**	**119**

4.2. Содержание дисциплины, структурированное по разделам и темам

Раздел 1. Основы разработки программных приложений.

1.1. Особенности анализа и проектирования крупных систем

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности разрабатываемых информационных систем. Для них характерны следующие особенности [14]:

- сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними) требует тщательного моделирования и анализа данных и процессов;
- наличие совокупности тесно взаимодействующих компонентов (подсистем);

- отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;
- необходимость интеграции существующих и вновь разрабатываемых подсистем;
- функционирование в неоднородной среде на разных аппаратных и операционных платформах;
- разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;
- существенная временная протяженность проекта, обусловленная ограниченными возможностями коллектива разработчиков, масштабами организации-заказчика, различной степенью готовности отдельных ее подразделений к внедрению информационных систем и т. д.;
- изменение или уточнение потребностей пользователей в процессе разработки и эксплуатации системы.

Непременным условием успешной реализации информационной системы является четкое и как можно более полное формирование требований на разработку системы, а также ее адекватное описание на стадии проектирования. Согласно [15]: «На обнаружение ошибок, допущенных на этапе анализа и проектирования, расходуется примерно в 2 раза больше времени, а на их исправление – примерно в 5 раз, чем на ошибки, допущенные на более поздних стадиях».

Основные требования на разработку информационной системы документально оформляются в виде календарного плана и технического задания. Детализация и реализация этих требований фиксируется в проектной документации.

1.2. Документы, содержащие требования на разработку системы

Основными документами, содержащими требования на разработку информационной системы, являются календарный план выполнения работ и техническое задание. Первый из них регламентирует состав, сроки и финансирование работ, а второй – основные требования к системе.

Пример календарного плана, являющегося обязательным приложением к договору, см. на рис. 4.1.

Приложение N 1
к договору № ____
от _____ 1999г.

УТВЕРЖДАЮ
Заместитель руководителя
Департамента управления перевозками МПС РФ
_____ В.В. Ипатов

КАЛЕНДАРНЫЙ ПЛАН
«Внедрение и сопровождение информационно-аналитической системы технико-экономических расчетов и вариантного анализа движения поездов в Департаменте управления перевозками и дорогах Центрального и Восточно-Сибирского регионов России»

№ этапа	Наименование основных этапов	Срок выполнения начало-окончание (число, м-ц, год)	Расчетная цена этапа, руб. / в % к договорной цене
1	2	3	4
1	Переработка и расширение системы плановых типово-энергетических расчетов в соответствии с новыми требованиями и предложениями дорог России.	01.01.99- 31.03.99	150000 25%
4	Внедрение и сопровождение системы в службах движения Северной и Восточно-Сибирской железных дорог. Организация и проведение курсов по системе (для Московской, Горьковской, Свердловской и Южно-Уральской дорог). Подготовка отчета.	01.10.99- 31.12.99	150000 25%
	Итого		600000

От «ИСПОЛНИТЕЛЯ»
Руководитель НИР
_____ В.А. Анисимов

От «ЗАКАЗЧИКА»
Начальник технического отдела
Департамента управления перевозками
_____ В.И. Малышев

Рис. 4.1. Пример календарного плана

Как было отмечено выше, техническое задание (ТЗ) является основным документом, определяющим требования и порядок создания (развития или модернизации) системы. Несмотря на то, что согласно [9, 13] ТЗ составляется после заключения договора, нередко оно должно быть подготовлено разработчиком еще до его подписания.

Состав, содержание, правила оформления этого документа устанавливаются ГОСТ 34.602–89 «Техническое задание на создание автоматизированной системы» [3]. ТЗ, как правило, содержит следующие разделы:

- общие сведения (наименование системы; наименование предприятий разработчика и заказчика с их реквизитами; перечень документов, на основании которых создается система, плановые сроки начала и окончания работы и т. д.);
- назначение и цели создания (развития) системы;
- характеристика объектов автоматизации;
- требования к системе в целом, к функциям и обеспечению. При этом выделяют следующие виды обеспечения:
  - *математическое* – совокупность математических методов, моделей и алгоритмов, применяемых в информационной системе;
  - *информационное* – совокупность форм документов, классификаторов, нормативной базы и реализованных решений по объемам, размещению и формам существования информации;
  - *лингвистическое* – совокупность правил применения в системе языков программирования, языков взаимодействия пользователей и технических средств системы, а также совокупность требований к кодированию и декодированию данных, к способам организации диалога и т. д.;
  - *программное* – совокупность программ и документов, предназначенных для отладки, функционирования и проверки работоспособности системы;
  - *техническое* – совокупность всех технических средств, используемых при функционировании системы;
  - *организационное* – совокупность документов, устанавливающих организационную структуру, права и обязанности пользователей и эксплуатационного персонала системы;
  - *методическое* – совокупность документов, описывающих технологию функционирования системы, методы выбора и применения пользователями технологических приемов для получения конкретных результатов;
- состав и содержание работ по созданию системы;
- порядок контроля и приемки системы;
- требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;
- требования к документированию;
- источники разработки – документы и информационные материалы (технико-экономическое обоснование, отчеты о законченных научно-исследовательских работах, информационные материалы на отечественные, зарубежные системы-аналоги и др.), на основании которых разрабатывалось ТЗ и которые должны быть использованы при создании системы.

Техническое задание может включать приложения, например:

- расчет ожидаемой эффективности системы;
- оценку научно-технического уровня системы;
- перечень основных входных и выходных форм и т. д.

Грамотно составленное ТЗ является залогом успешной реализации проекта. В некоторых организациях, обычно имевших «печальный» опыт недооценки этого важного документа, существует практика составления расширенного и более детального технического задания. Особое внимание при его подготовке следует уделить полному и точному описанию

требований, полный перечень которых приведен в [3]. В противном случае разработчик может создать систему, которая не нужна заказчику или не соответствует всем его ожиданиям.

Наиболее полно и точно должны быть определены требования к системе в целом (требования к структуре и функционированию системы; к численности и квалификации персонала системы и режиму его работы; к защите информации от несанкционированного доступа; к патентной чистоте и др.) и ее функциям (требования к составу и назначению подсистем; к режимам функционирования системы и т. д.). Заказчик и разработчик должны определиться, какие функции будет выполнять система (какие задачи и как она будет решать), а какие – не будет.

Раздел 2. Технологии и подходы к проектированию программных приложений

2.1. CASE-технологии анализа и проектирования

Максимально упростить и формализовать процессы формирования требований и проектирования системы позволяют современные CASE-средства.

В 70-х и 80-х гг. XX в. при разработке информационных систем достаточно широко стала применяться структурная методология анализа, предоставляющая в распоряжение разработчиков строгие формализованные методы описания системы и принимаемых технических решений. Она основана на применении наглядной графической техники (схем и диаграмм), предназначенной для описания различного рода моделей. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических (проектных) решений [14].

Перечисленные факторы способствовали появлению специальных программных средств – CASE-средств, реализующих CASE-технологию создания и сопровождения информационных систем. Термин *CASE* используется в настоящее время весьма широко. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения, в настоящее время приобрело новый смысл, охватывающий процесс разработки и сопровождения сложных систем в целом.

CASE-технология представляет собой методологию проектирования информационных систем, набор методов, нотаций[1] и инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать модель системы на всех этапах разработки и сопровождения системы и разрабатывать приложения в соответствии с информационными потребностями пользователей [14].

В качестве инструментария реализации технологии используются CASE-средства, основными функциями которых являются:

- централизованное хранение в единой базе данных проекта (репозитарии) информации об информационной системе в течение всего жизненного цикла. Репозитарий может хранить объекты различных типов: диаграммы, определения экранов и меню, проекты отчетов, описание данных, логику их обработки, исходные коды программ и т.п.;
- прямое проектирование программного обеспечения и баз данных. При этом порядок использования разработчиками CASE-средства следующий:
  - создается логическая модель системы;
  - выбирается конкретный язык программирования или СУБД для построения физической модели, после чего CASE-средство автоматически создает физическую модель системы;
  - дорабатывается физическая модель;
  - выполняется автоматическая генерация текста программы или структуры базы данных на диске;
- обратное проектирование (реинжиниринг). В этом случае порядок использования CASE-средства обратный – от текста программы или базы данных на диске к логической модели. Помимо построения, CASE-средства позволяют быстро интегрировать полученные таким образом модели в проект, а также с меньшими потерями переходить от одной физической реализации к другой (например, в случае ухода «старых» разработчиков, плохо документирующих программное обеспечение, или появления новых, более перспективных языков программирования и СУБД);
- синхронизация моделей системы с ее физической реализацией. В случае изменения модели системы могут быть автоматически внесены необходимые изменения в физическую реализацию или наоборот;
- автоматическое обеспечение качества и тестирование моделей на наличие ошибок (например, ошибок нормализации БД), полноту и непротиворечивость;
- автоматическая генерация документации. Вся документация по проекту генерируется автоматически на базе репозитария (как правило, в соответствии с требованиями действующих стандартов). Несомненное достоинство CASE-технологии заключается в том, что документация всегда отвечает текущему состоянию дел, поскольку любые изменения в проекте автоматически отражаются в репозитарии.

2.2 Сущность структурного анализа и проектирования

Программы, написанные на первых языках программирования, как и первая программа, написанная лично, представляли собой простое перечисление команд, выполняемых от начала до конца. Естественно, в этих программах уже присутствовали управляющие конструкции типа циклов, условных и безусловных переходов и т. д. Но по мере усложнения программ (увеличения кода) их разработка и сопровождение при таком подходе становились все более затруднительными. Тогда в языках стали появляться дополнительные конструкции (функции и процедуры), возможность модульной разработки программы и последующая ее сборка из разных файлов. Программа стала представлять собой не просто единое целое с трудно различимым внутренним устройством (типа «черного ящика»), а структуру, состоящую из четко выраженных модулей, связанных между собой определенными отношениями (интерфейсами). То есть *программа приобрела структуру иерархической многоуровневой модульной системы*. Каждый уровень такой системы является законченным модулем, поддерживаемым и контролируемым модулем, находящимся над ним.

Методологии структурного анализа и проектирования информационных систем появились позже фактического использования этих принципов на практике (структурного программирования). В конце 60-х гг. XX в. стали появляться и применяться первые методологии, ориентированные на структурный подход.

Таблица 5.1. Методологии структурного анализа и проектирования

Методология	Тип разрабатываемой модели
SADT (Structured Analysis and Design Technique, методология структурного анализа и проектирования)	**Функциональная**
DFD (Data Flow Diagrams, диаграммы потоков данных)	**Смешанная** (функциональная, информационная, компонентная)

ERD (Entity-Relationship Diagrams, диаграммы «сущность-связь»)	**Информационная**
STD (State Transition Diagrams, диаграммы изменения состояний)	**Поведенческая**
Flowcharts (блок-схемы)	**Смешанная** (поведенческая, информационная, компонентная)

Схематично применение структурного подхода изображено на рис. 5.1.

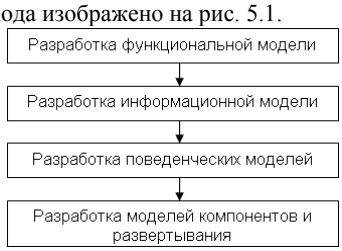

Рис. 5.1. Схема применения структурного подхода

В начале разрабатывается функциональная модель, с помощью которой определяются, анализируются и фиксируются требования к составу и структуре функций системы, т. е. определяется, для каких целей разрабатывается система, какие функции она будет выполнять. На этой же модели указываются исходная информация, промежуточные и итоговые результаты работы системы. На основе информационных потоков определяется состав и структура необходимых данных, хранимых в системе (строится информационная модель). Далее, с учетом разработанных моделей, создаются процедуры реализации функции, т. е. алгоритмы обработки данных и поведения элементов системы. На заключительной стадии устанавливается распределение функций по подсистемам (компонентам), необходимое техническое обеспечение и строится модель их распределения по узлам системы.

Показанная на рис. 5.1 схема не означает, что построение моделей должно строго соответствовать указанному порядку – одна за другой. Как правило, разработка каждой последующей модели начинается еще до полного завершения разработки предыдущей, а иногда – параллельно.

На стадии проектирования модели расширяются, уточняются и дополняются диаграммами, отражающими технологию использования системы, ее архитектуру, экранные формы и т. п.

2.3. РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ МОДЕЛИ

2.3.1. Основы функционального анализа и проектирования систем

Перед разработкой системы заказчик и разработчик должны ясно представлять, какие функциональные возможности будут заложены в систему и как будет организовано функциональное взаимодействие внутри системы.

При разработке функциональной модели (определении функциональных требований) может возникнуть множество проблем:

- заказчик не может точно выразить, решение каких задач возлагается на информационную систему. Зачастую заказчик даже не знает, что такое требование и как его формулировать;
- представители заказчика (начальники разных уровней, эксперты-технологи, рядовые пользователи) по-своему видят работу будущей системы и часто их требования к системе носят взаимоисключающий характер. Особенно характерна такая ситуация, когда разрабатываемая система будет внедряться на нескольких объектах автоматизации;
- заказчик зачастую не знает возможностей современных вычислительных систем и стремится рассматривать процесс автоматизации как простой перенос элементарных видов деятельности, выполняемых вручную, на компьютеры. При этом он не задумывается об оптимизации бизнес-процессов внутри организации с приходом новых технологий;
- заказчик не верит в возможность выполнения некоторых функций «бездушными» машинами.

Построение функциональной модели должно решить большую часть этих проблем. Наиболее подробно этот процесс рассмотрен в [18, 19].

При ее разработке сначала строится модель существующей организации работы AS-IS (как есть) на основе должностных инструкций, приказов, отчетов, нормативной документации и т. д. Она позволяет выяснить, «что мы делаем сегодня» перед тем, как «перепрыгнуть» на то, «что мы будем делать завтра» [18, 19]. Анализ модели позволяет понять, где находятся слабые места, в чем будут состоять преимущества новых процессов и насколько глубоким изменениям подвергнется существующая организация деятельности предприятия (компании, отдела). Признаками неэффективной организации деятельности могут быть:

- бесполезные, неуправляемые и дублирующие работы;
- работы без результата;
- неэффективный документооборот (нужный документ не оказывается в нужное время в нужном месте) и т. д.

2.3.2. Назначение и состав методологии SADT (IDEF0)

Методология SADT (Structured Analysis and Design Technique – методология структурного анализа и проектирования) представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели системы.

Начало разработки данной методологии было положено Дугласом Россом (США) в середине 60-х гг. XX в. С тех пор системные аналитики компании SofTech, Inc. улучшили SADT и использовали ее в решении широкого круга проблем. Программное обеспечение телефонных сетей, диагностика, долгосрочное и стратегическое планирование, автоматизированное производство и проектирование, конфигурация компьютерных систем, обучение персонала, управление финансами и материально-техническим снабжением – вот некоторые из областей эффективного применения SADT. Широкий спектр областей указывает на универсальность и мощь методологии SADT. В программе «Интеграции компьютерных и промышленных технологий» (Integrated Computer Aided Manufacturing, ICAM) Министерства обороны США была признана полезность SADT. Это привело к публикации ее части в 1981 г., называемой *IDEF0* (Icam DEFinition), в качестве федерального стандарта на разработку программного обеспечения. Под этим названием SADT стала применяться тысячами специалистов в военных и промышленных организациях [15]. Последняя редакция стандарта IDEF0 была выпущена в декабре 1993 г. Национальным институтом по стандартам и технологиям США (National Institute Standards and Technology, NIST).

Стоит отметить, что IDEF0 рекомендована для использования Госстандартом РФ и активно применяется в отечественных госструктурах (например, в Государственной налоговой инспекции РФ).

Данная методология при описании функционального аспекта информационной системы конкурирует с методами, ориентированными на потоки данных (DFD). В отличие от них IDEF0 позволяет:

- описывать любые системы, а не только информационные (DFD предназначена для описания программного обеспечения);
- создать описание системы и ее внешнего окружения до определения окончательных требований к ней. Иными словами, с помощью данной методологии можно постепенно выстраивать и анализировать систему даже тогда, когда трудно еще представить ее воплощение.

Таким образом, IDEF0 может применяться на ранних этапах создания широкого круга систем. В то же время она может быть использована для анализа функций существующих систем и выработки решений по их улучшению.

Основу методологии IDEF0 составляет графический язык описания процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Модель (AS-IS, TO-BE или SHOULD-BE) может содержать *4 типа диаграмм* [14, 15, 18, 19]:

- контекстную диаграмму;
- диаграммы декомпозиции;
- диаграммы дерева узлов;
- диаграммы только для экспозиции (for exposition only, FEO).

Контекстная диаграмма (диаграмма верхнего уровня), являясь вершиной древовидной структуры диаграмм, показывает назначение системы (основную функцию) и ее взаимодействие с внешней средой. В каждой модели может быть только одна контекстная диаграмма. После описания основной функции выполняется функциональная декомпозиция, т. е. определяются функции, из которых состоит основная.

Далее функции делятся на подфункции и так до достижения требуемого уровня детализации исследуемой системы. Диаграммы, которые описывают каждый такой фрагмент системы, называются *диаграммами декомпозиции*. После каждого сеанса декомпозиции проводятся сеансы экспертизы – эксперты предметной области указывают на соответствие реальных процессов созданным диаграммам. Найденные несоответствия устраняются, после чего приступают к дальнейшей детализации процессов.

2.3.3. Элементы графической нотации IDEF0

Методология IDEF0 нашла широкое признание и применение, в первую очередь, благодаря простой графической нотации, используемой для построения модели. Главными компонентами модели являются диаграммы. На них отображаются функции системы в виде прямоугольников, а также связи между ними и внешней средой посредством стрелок. Использование всего лишь двух графических примитивов (прямоугольник и стрелка) позволяют быстро объяснить правила и принципы построения диаграмм IDEF0 людям, незнакомым с данной методологией. Это достоинство позволяет подключить и активизировать деятельность заказчика по описанию бизнес-процессов с использованием формального и наглядного графического языка.

Рассмотрим основные элементы графической нотации IDEF0 (рис. 6.1.) [15, 18].

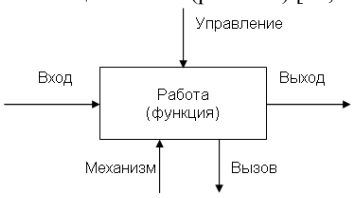

Рис. 6.1. Элементы графической нотации IDEF0

Прямоугольник представляет собой *работу (процесс, деятельность, функцию или задачу)*, которая имеет фиксированную цель и приводит к некоторому конечному результату. Имя работы должно выражать действие (например, «Изготовление детали», «Расчет допускаемых скоростей», «Формирование ведомости ЦДЛ № 3»).

Взаимодействие работ между собой и внешним миром описывается в виде стрелок. В IDEF0 различают *5 видов стрелок*:

- *вход* (англ. input) – материал или информация, которые используются и преобразуются работой для получения результата (выхода). Вход отвечает на вопрос «Что подлежит обработке?». В качестве входа может быть как материальный объект (сырье, деталь, экзаменационный билет), так и не имеющий четких физических контуров (запрос к БД, вопрос преподавателя). Допускается, что работа может не иметь ни одной стрелки входа. Стрелки входа всегда рисуются входящими в левую грань работы;
- *управление* (англ. control) – управляющие, регламентирующие и нормативные данные, которыми руководствуется работа. Управление отвечает на вопрос «В соответствии с чем выполняется работа?». Управление влияет на работу, но не преобразуется ей, т. е. выступает в качестве ограничения. В качестве управления могут быть правила, стандарты, нормативы, расценки, устные указания. Стрелки управления рисуются входящими в верхнюю грань работы. Если при построении диаграммы возникает вопрос, как правильно нарисовать стрелку сверху или слева, то рекомендуется ее рисовать как вход (стрелка слева);
- *выход* (англ. output) – материал или информация, которые представляют результат выполнения работы. Выход отвечает на вопрос «Что является результатом работы?». В качестве выхода может быть как материальный объект (деталь, автомобиль, платежные документы, ведомость), так и нематериальный (выборка данных из БД, ответ на вопрос, устное указание). Стрелки выхода рисуются исходящими из правой грани работы;
- *механизм* (англ. mechanism) – ресурсы, которые выполняют работу. Механизм отвечает на вопрос «Кто выполняет работу или посредством чего?». В качестве механизма могут быть персонал предприятия, студент, станок, оборудование, программа. Стрелки механизма рисуются входящими в нижнюю грань работы;
- *вызов* (англ. call) – стрелка указывает, что некоторая часть работы выполняется за пределами рассматриваемого блока. Стрелки выхода рисуются исходящими из нижней грани работы

2.3.4. Типы связей между работами

После определения состава функций и взаимосвязей между ними, возникает вопрос о правильной их композиции (объединении) в модули (подсистемы). При этом подразумевается, что каждая отдельная функция должна решать одну, строго определенную задачу. В противном случае необходима дальнейшая декомпозиция или разделение функций.

При объединении функций в подсистемы необходимо стремиться, чтобы внутренняя связность (между функциями внутри модуля) была как можно сильнее, а внешняя (между функциями, входящими в разные модули), как можно слабее. Опираясь на семантику связей методологии IDEF0, введем классификацию связей между функциями (работами). Данная классификация является расширением [14]. Типы связей приводятся в порядке уменьшения их значимости (силы связывания). В приводимых примерах утолщенными линиями выделяются функции, между которыми имеется рассматриваемый тип связи.

1. *Иерархическая связь (связь «часть» – «целое»)* имеет место между функцией и подфункциями, из которых она состоит

2. *Регламентирующая (управляющая, подчиненная) связь* отражает зависимость одной функции от другой, когда выход одной работы направляется на управление другой. Функцию, из которой выходит управление, следует считать регламентирующей или управляющей, а в которую входит – подчиненной. Различают *прямую связь по управлению*, когда управление передается с вышестоящей работы на нижестоящую (рис. 6.3), и *обратную связь по управлению*, когда управление передается от нижестоящей к вышестоящей

3. *Функциональная (технологическая) связь* имеет место, когда выход одной функции служит входными данными для следующей функции. С точки зрения потока материальных объектов данная связь показывает технологию (последовательность работ) обработки этих объектов. Различают *прямую связь по входу*, когда выход передается с вышестоящей работы на нижестоящую (рис. 6.5), и *обратную связь по входу*, когда выход передается с нижестоящей к вышестоящей

4. *Потребительская связь* имеет место, когда выход одной функции служит механизмом для следующей функции. Таким образом, одна функция потребляет ресурсы, вырабатываемые

5. *Логическая связь* наблюдается между логически однородными функциями . Такие функции, как правило, выполняют одну и ту же работу, но разными (альтернативными) способами или, используя разные исходные данные (материалы).

6. *Коллегиальная (методическая) связь* имеет место между функциями, алгоритм работы которых определяется одним и тем же управлением (рис. 6.9). Аналогом такой связи является совместная работа сотрудников одного отдела (коллег), подчиняющихся начальнику, который отдает указания и приказы (управляющие сигналы). Такая связь также возникает, когда алгоритмы работы этих функций определяются одним и тем же методическим обеспечением (СНИП, ГОСТ, официальными нормативными материалами и т. д.), служащим в качестве управления.

7. *Ресурсная связь* возникает между функциями, использующими для своей работы одни и те же ресурсы (рис. 6.10). Ресурсно-зависимые функции, как правило, не могут выполняться одновременно.

8. *Информационная связь* имеет место между функциями, использующими в качестве входных данных одну и ту же информацию

9. *Временная связь* возникает между функциями, которые должны выполняться одновременно до или одновременно после другой функции (рис. 6.12).

Кроме указанных на рис. 19 случаев, эта связь имеет место также между другими сочетаниями управления, входа и механизма, поступающими в одну функцию.

10. *Случайная связь* возникает, когда конкретная связь между функциями мала или полностью отсутствует

Из приведенных выше типов связей наиболее сильной является иерархическая связь, которая, по сути, и определяет объединение функций в модули (подсистемы). Несколько слабее являются регламентирующие, функциональные и потребительские связи. Функции с этими связями обычно реализуются в одной подсистеме. Логические, коллегиальные, ресурсные и информационные связи одни из самых слабых. Функции, обладающие ими, как правило, реализуют в разных подсистемах, за исключением логически однородных функций (функций, связанных логической связью). Временная связь свидетельствует о слабой зависимости функций друг от друга и требует их реализации в отдельных модулях.

Таким образом, при объединении функций в модули наиболее желательными являются первые пять видов связей. Функции, связанные последними пятью связями, лучше реализовывать в отдельных модулях.

2.3.5. Правила и рекомендации построения диаграмм IDEF0

В IDEF0 существуют соглашения (правила и рекомендации) по созданию диаграмм, которые призваны облегчить чтение и экспертизу модели [14, 15, 18, 19]. Некоторые из этих правил CASE-средства поддерживают автоматически, выполнение других следует обеспечить вручную.

1. Перед построением модели необходимо определиться, какая модель (модели) системы будет построена. Это подразумевает определение ее типа AS-IS, TO-BE или SHOULD-BE, а также определения позиции, с точки зрения которой строится модель. «Точку зрения» лучше всего представлять себе как место (позицию) человека или объекта, в которое надо встать, чтобы увидеть систему в действии. Например, при построении модели работы продуктового магазина можно среди возможных претендентов, с точки зрения которых рассматривается система, выбрать продавца, кассира, бухгалтера или директора. Обычно выбирается одна точка зрения, наиболее полно охватывающая все нюансы работы системы, и при необходимости для некоторых диаграмм декомпозиции строятся диаграммы FEO, отображающие альтернативную точку зрения.

2. На контекстной диаграмме отображается один блок, показывающий назначение системы. Для него рекомендуется отображать по

2–4 стрелки, входящие и выходящие с каждой стороны.

3. Количество блоков на диаграммах декомпозиции рекомендуется в пределах 3–6. Если на диаграмме декомпозиции два блока, то она, как правило, не имеет смысла. При наличии большого количества блоков диаграмма становится перенасыщенной и трудно читаемой.

4. Блоки на диаграмме декомпозиции следует располагать слева направо и сверху вниз. Такое расположение позволяет более четко отразить логику и последовательность выполнения работ. Кроме этого маршруты стрелок будут менее запутанными и иметь минимальное количество пересечений.

5. Отсутствие у функции одновременно стрелок управления и входа не допускается. Это означает, что запуск данной функции не контролируется и может произойти в любой произвольный момент времени либо вообще никогда

2.3.6. Пример построения модели IDEF0 для системы определения допускаемых скоростей

Расчет допускаемых скоростей движения поездов является трудоемкой инженерной задачей. При проходе поездом какого-либо участка фактическая скорость движения поезда не должна превышать предельно допускаемую. Эта предельно допускаемая скорость устанавливается исходя из опыта эксплуатации и специально проводимых испытаний по динамике движения и воздействию на путь подвижного состава. Непревышение этой скорости гарантирует безопасность движения поездов, комфортабельные условия езды пассажиров и т. п. Они определяются в зависимости от типа подвижного состава (марки локомотива и типа вагонов), параметров верхнего строения пути (типа рельсов, балласта, эпюры шпал) и плана

(радиуса кривых, переходных кривых, возвышения наружного рельса и т. д.). Как правило, для установления допускаемых скоростей необходимо определить не менее двух (на прямых) и пяти (в кривых) скоростей, из которых и выбирается окончательная допускаемая скорость, как наименьшая из всех рассчитанных. Расчет этих скоростей регламентируются Приказом МПС России № 41 от 12 ноября 2001 г. «Нормы допускаемых скоростей движения подвижного состава по железнодорожным путям колеи 1520 (1524) мм Федерального железнодорожного транспорта».

Как было отмечено, построение модели IDEF0 начинается с представления всей системы в виде простейшей компоненты (контекстной диаграммы). Данная диаграмма отображает назначение (основную функцию) системы и необходимые входные и выходные данные, управляющую и регламентирующую информацию, а также механизмы.

Контекстная диаграмма для задачи определения допускаемых скоростей показана на рис.6.21. Для построения модели использовался продукт BPwin 4.0 фирмы Computer Associates.

В качестве *исходной информации*, на основе которой выполняется определение допускаемых скоростей, используются:

- данные проекта новой линии или проекта реконструкции (содержат всю необходимую информацию для реализации проекта, а именно километраж, оси раздельных пунктов, план линии и др.);
- подробный продольный профиль (содержит информацию, аналогичную рассмотренной выше);
- паспорт дистанции пути (содержит информацию, аналогичную рассмотренной выше, а также сведения о верхнем строении пути (ВСП));
- данные о результатах съемки плана пути вагоном-путеизмерителем;
- ведомость возвышений наружного рельса в кривых (содержит информацию о плане пути).

Часть исходной информации может быть взята из разных источников. В частности сведения о плане (параметрах кривых) могут быть взяты из проекта новой линии или проекта реконструкции, подробного продольного профиля, паспорта дистанции пути и т. д.

Управляющими данными являются:

- указание начальника службы пути дороги или Департамента пути и сооружений ОАО «РЖД» на расчет;
- Приказ № 41, содержащий нормативно-справочную информацию, порядок и формулы определения допускаемых скоростей;
- сведения о текущем или планируемом поездопотоке (данные о марках обращающихся локомотивов и типах используемых вагонов);
- сведения о планируемых ремонтах пути, реконструкции и переустройстве сооружений и устройств.

Результатом работы системы должны быть:

- ведомости допускаемых скоростей, содержащие все типы рассчитанных скоростей и позволяющие установить причину их ограничения;
- ведомости Приказа начальника дороги об установлении допускаемых скоростей на перегонах и раздельных пунктах (Приказ «Н») согласно принятой на дороге форме. Утвержденный Приказ «Н» официально закрепляет допускаемые скорости движения поездов;
- типовые формы № 1, 1а и 2, содержащие планируемые допускаемые скорости для разработки графика движения поездов.

Скорости, содержащиеся в Приказе «Н» и типовых формах, могут отличаться от рассчитанных и показываемых в ведомостях допускаемых скоростей. Это связано с тем, что в них отражают ограничения скорости не только по конструкции подвижного состава, параметров ВСП и кривых, но и по состоянию устройств и сооружений (деформация земляного полотна, перекос опор контактной сети и т. д.). Кроме того, они корректируются с учетом планируемых ремонтов пути, реконструкции и переустройства сооружений и устройств и т. д.

После построения контекстная диаграмма детализируется с помощью диаграммы декомпозиции первого уровня. На этой диаграмме отображаются функции системы, которые должны быть реализованы в рамках основной функции. Диаграмма, для которой выполнена декомпозиция, по отношению к детализирующим ее диаграммам называется *родительской*. Диаграмма декомпозиции по отношению к родительской называется *дочерней*.

2.3.7. ICOM-коды

Стрелки, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются теми же самыми, что и стрелки, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы (см. рис. 6.21 и 6.22). Как следствие этого, границы функции верхнего уровня – это то же самое, что и границы диаграммы декомпозиции.

ICOM-коды (аббревиатура от Input, Control, Output и Mechanism) предназначены для идентификации граничных стрелок. ICOM-код содержит префикс, соответствующий типу стрелки (I, C, O или M), и порядковый номер (см. рис. 6.22).

2.3.8. Назначение и состав DFD

При построении функциональной модели системы альтернативой методологии SADT (IDEF0) является методология *диаграмм потоков данных* (Data Flow Diagrams, DFD). В отличие от IDEF0, предназначенной для проектирования систем вообще, DFD предназначена для проектирования информационных систем. Ориентированность этой методологии на проектирование автоматизированных систем делает ее удобным и более выгодным инструментом при построении функциональной модели ТО-ВЕ.

Как и в IDEF0 основу методологии DFD составляет графический язык описания процессов. Авторами одной из первых графических нотаций DFD (1979 г.) стали Эд Йордан (Yourdon) и Том де Марко (DeMarko).

В настоящее время наиболее распространенной является нотация Гейна-Сарсона (Gane-Sarson).

Модель системы в нотации DFD представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе. Модель системы содержит контекстную диаграмму и диаграммы декомпозиции.

Принципы построения функциональной модели с помощью DFD аналогичны принципам методологии IDEF0. Вначале строится контекстная диаграмма, где отображаются связи системы с внешним окружением. В дальнейшем выполняется декомпозиция основных процессов и подсистем с построением иерархии диаграмм.

2.3.9. Элементы графической нотации DFD

Согласно DFD источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям – потребителям информации [14, 17, 18, 19].

Поток данных определяет информацию (материальный объект), передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т. д.

Каждый поток данных имеет имя, отражающее его содержание. Направление стрелки показывает направление потока данных. Иногда информация может двигаться в одном направлении, обрабатываться и возвращаться назад в ее источник. Такая ситуация может моделироваться либо двумя различными потоками, либо одним – двунаправленным.

На диаграммах IDEF0 потоки данных соответствуют входам и выходам, но в отличие от IDEF0 стрелки потоков на DFD могут отображаться входящими и выходящими из любой грани внешней сущности, процесса или накопителя данных.

Процесс (в IDEF0 – функция, работа) представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом.

Каждый процесс должен иметь имя в виде предложения с глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже, например:

· «Ввести сведения о клиентах»;

· «Рассчитать допускаемую скорость»;

· «Сформировать ведомость допускаемых скоростей»

Номер процесса служит для его идентификации и ставится с учетом декомпозиции. В отличие от IDEF0 вложенность процессов обозначается через точку (например, в IDEF0 – «236», в DFD – «2.3.6»).

Преобразование информации может показываться как с точки зрения процессов, так и с точки зрения *систем и подсистем*. Если вместо имени процесса «Рассчитать допускаемую скорость» написать «Подсистема расчета допускаемых скоростей», тогда этот блок на диаграмме стоит рассматривать, как подсистему.

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

Накопитель данных может быть реализован физически в виде ящика в картотеке, области в оперативной памяти, файла на магнитном носителе и т. д.

Накопителю обязательно должно даваться уникальное имя и номер в пределах всей модели (всего набора диаграмм). Имя накопителя выбирается из соображения наибольшей информативности для разработчика. Например, если в качестве накопителей выступают таблицы проектируемой базы данных, тогда в качестве имен накопителей рекомендуется использовать имена таблиц. Таким образом, накопитель данных может представлять собой всю базу данных целиком, совокупность таблиц или отдельную таблицу. Такое представление накопителей в дальнейшем облегчит построение информационной модели системы.

2.3.10. Правила и рекомендации построения DFD

Правила и рекомендации построения модели DFD в основном совпадают с принятыми в IDEF0. Часть из них приведена в подразд. 6.9.

По аналогии с IDEF0 у каждого процесса (подсистемы) на диаграмме потоков данных должен быть как минимум один входящий и один выходящий поток. Процесс должен запускаться на выполнение либо через обрабатываемый, либо через управляющий поток данных. Работа каждого процесса должна завершаться конкретным результатом.

Каждый накопитель данных также должен иметь как минимум один входящий и один выходящий поток. Наличие только входящих потоков в накопитель означает, что информация накапливается, но не используется.

Наличие только выходящих потоков из накопителя также является ошибкой. Прежде чем использовать данные из накопителя, они должны там появиться в результате работы какого-либо процесса (подсистемы, внешней сущности). Исключением из правил считается случай, когда накопитель является внешней сущностью. Тогда допускается наличие либо только входящих стрелок, либо только выходящих стрелок (см. рис. 6.23, накопитель «БД АРМ-П или СБД-П»).

2.3.11. Пример построения модели DFD для системы определения допускаемых скоростей

Описание задачи приведено в подразд. 6.6.

Построение функциональной модели DFD начинается как и в IDEF0 с разработки контекстной диаграммы. На ней отображается основной процесс (сама система в целом) и ее связи с внешней средой (внешними сущностями). Это взаимодействие показывается через потоки данных. Допускается на контекстной диаграмме отображать сразу несколько основных процессов или подсистем.

На этой диаграмме видно, что в качестве источника исходных данных для работы системы могут использоваться базы данных АРМ-П (АРМ службы пути) или СБД-П (Сводная БД – Путейский фрагмент), содержащие практически всю необходимую информацию по участкам дороги.

В то же время в системе оставлена возможность ее ручного ввода и корректировки. Несмотря на то, что БД АРМ-П или СБД-П по отношению к системе являются внешними сущностями, они, в целях лучшего восприятия, показаны в виде накопителя данных.

Дальнейший процесс проектирования состоит в построении диаграмм декомпозиции, которые строятся (показывают устройство) *только для процессов или подсистем (систем).*

На этом рисунке у некоторых потоков данных, связанных с накопителями, отсутствуют имена. Это позволяет устранить дублирование надписей и, как следствие, уменьшить насыщенность диаграммы.

При построении диаграммы декомпозиции блоки системы в одних случаях показаны как процессы (имя начинается с глагола), в других – как подсистемы (имя начинается со слова «подсистема»). Это сделано в целях иллюстрации правил именования блоков. В то же время декомпозицию системы можно было бы представить, либо используя только процессы, либо только подсистемы.

Контекстная диаграмма и диаграмма декомпозиции выполнены с использованием BPwin 4.0.

Решение о завершении детализации процесса и использовании миниспецификации принимается проектировщиком исходя из следующих критериев:

- наличия у процесса относительно небольшого количества входных и выходных потоков данных (2–3 потока);
- возможности описания процессов в виде простого алгоритма;
- возможности описания логики процесса при помощи миниспецификации небольшого объема (не более 20–30 строк).

2.3.12. Расширения DFD для систем реального времени

Системы реального времени построены, как правило, на взаимодействии средств вычислительной техники и различных физических устройств съема информации (датчиков, камер, микрофонов и т. д.). Первые являются дискретными

преобразователями информации, вторые в основном – аналоговыми, т. е. генерирующими информацию в виде непрерывного потока. Другой особенностью таких систем является значительный уклон в сторону управления объектами. Для моделирования особенностей поведения систем реального времени П. Вард и С. Меллор предложили использовать на DFD дополнительные элементы.

Квазинепрерывный поток (лат. quasi – как будто, якобы) – поток данных, непрерывный во времени. Отображается линией с двумя стрелками на

Управляющий процесс – процесс, формирующий сигналы управления на

Управляющий поток – управляющая информация, запускающая процесс (подсистему) или изменяющая ход его выполнения

Использование управляющих потоков позволяет отделить управляющую информацию от обрабатываемой, как это делается на диаграммах IDEF0.

Накопитель управлений – накопитель управляющих потоков

Раздел 3. Разработка информационной модели

3.1. Основы проектирования баз данных

Разработанная функциональная модель системы отвечает на вопросы «Что должна делать система?» и «За счет каких действий может быть достигнут требуемый результат?». Эта модель также позволяет концептуально определить наборы данных, используемых в системе.

В то же время она не отвечает на вопрос «Каким образом организованы данные в системе?». Для ответа на него необходимо построить информационную модель (запроектировать БД).

Традиционно процедуру проектирования базы данных разбивают на три этапа, каждый из которых завершается созданием соответствующей информационной модели [1, 20, 21].

Этап 1-й. Концептуальное проектирование – создание представления (схемы, модели) БД, включающего определение важнейших сущностей (таблиц) и связей между ними, но не зависящего от модели БД (иерархической, сетевой, реляционной и т. д.) и физической реализации (целевой СУБД).

Этап 2-й. Логическое проектирование – развитие концептуального представления БД с учетом принимаемой модели (иерархической, сетевой, реляционной и т.д.).

Этап 3-й. Физическое проектирование – развитие логической модели БД с учетом выбранной целевой СУБД.

Концептуальное и логическое проектирование вместе называют также *инфологическим* или *семантическим проектированием*.

В настоящее время для проектирования БД активно используются CASE-средства, в основном ориентированные на использование *ERD (Entity – Relationship Diagrams, диаграммы «сущность–связь»)*. С их помощью определяются важные для предметной области объекты (сущности), отношения друг с другом (связи) и их свойства (атрибуты). Следует отметить, что средства проектирования ERD в основном ориентированы на реляционные базы данных (РБД), и если существует необходимость проектирования другой системы, скажем объектно-ориентированной, то лучше избрать другие методы проектирования.

ERD были впервые предложены П. Ченом в 1976 г. Основные элементы ERD перечислены ниже [1, 18–21].

Сущность (таблица, в РБД – отношение) – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению. Если выражаться точнее, то это не объект, а набор объектов (класс) с одинаковыми свойствами. Примеры сущностей: работник, деталь, ведомость, результаты сдачи экзамена и т. д.

Экземпляр сущности (запись, строка, в РБД – кортеж) – уникально идентифицируемый объект.

Связь – некоторая ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Примерами связей могут являться родственные отношения «отец–сын», производственные – «начальник-подчиненный» или произвольные – «иметь в собственности», «обладать свойством».

Атрибут (столбец, поле) – свойство сущности или связи.

3.2. Концептуальное проектирование с использованием методологии IDEF1X

Цель концептуального проектирования – создание концептуальной модели данных на основе представлений о предметной области каждого отдельного типа пользователей. *Концептуальная модель* представляет собой описание основных сущностей (таблиц) и связей между ними без учета принятой модели БД и синтаксиса целевой СУБД. Часто на такой модели отображаются только имена сущностей (таблиц) без указания их атрибутов. *Представление пользователя* включает в себя данные, необходимые конкретному пользователю для принятия решений или выполнения некоторого задания.

Ниже рассматривается последовательность шагов при концептуальном проектировании. *Выделение сущностей.*

Первый шаг в построении концептуальной модели данных состоит в определении основных объектов (сущностей), которые могут интересовать пользователя и, следовательно, должны храниться в БД. При наличии функциональной модели IDEF0 прообразами таких объектов являются входы, управления и выходы. Еще лучше для этих целей использовать DFD. Прообразами объектов в этом случае будут накопители данных. Как было отмечено выше, накопитель данных является совокупностью таблиц (набором объектов) или непосредственно таблицей (объектом). Для более детального определения набора основных объектов необходимо также проанализировать потоки данных и весь методический материал, требуемый для решения задачи. Например, для задачи определения допускаемых скоростей основными объектами (наборами объектов) являются: нормативно-справочная информация, информация об участках дороги, задания на расчет, ведомости допускаемых скоростей и т. д. В ходе анализа и проектирования информационной модели наборы объектов должны быть детализированы. Например, составной объект «информация об участках дороги» с учетом специфики решаемой задачи требует разбиения на отдельные составляющие: участки, пути, раздельные пункты, километраж, план, верхнее строение пути и т. д.

Возможные трудности в определении объектов связаны с использованием постановщиками задачи:

· примеров и аналогий при описании объектов (например, вместо обобщающего понятия «работник» они могут упоминать его функции или занимаемую должность: «руководитель», «ответственный», «контролер», «заместитель»);

· синонимов (например, «допускаемая скорость» и «установленная скорость», «разработка» и «проект», «барьерное место» и «ограничение скорости»);

· омонимов (например, «программа» может обозначать компьютерную программу, план предстоящей работы или программу телепередач).

Далеко не всегда очевидно то, чем является определенный объект – сущностью, связью или атрибутом. Например, как следует классифицировать «семейный брак»? На практике это понятие можно вполне обоснованно отнести к любой из упомянутых категорий. Анализ является субъективным процессом, поэтому различные разработчики могут создавать

разные, но вполне допустимые интерпретации одного и того же факта. Выбор варианта в значительной степени зависит от здравого смысла и опыта проектировщика.

2. Определение атрибутов.

3.3. Логическое проектирование с использованием методологии IDEF1X

Цель логического проектирования – развить концептуальное представление БД с учетом принимаемой модели БД (иерархической, сетевой, реляционной и т. д.).

Примем в качестве модели реляционную БД в третьей нормальной форме (набор нормализованных отношений с кратностью связей 1:N). Поэтому необходимо будет проверить концептуальную модель с помощью методов нормализации и контроля выполнения транзакций [1, 20, 21]. *Транзакция* – одно действие или их последовательность, выполняемых как единое целое одним или несколькими пользователями (прикладными программами) с целью осуществления доступа к БД и изменению ее содержимого.

1. Удаление и проверка элементов, не отвечающих принятой модели данных.

1.1. Удаление связей N:M.

Если в концептуальной модели присутствуют связи N:M, то их следует устранить путем определения промежуточной сущности. Связь N:M заменяется двумя

1.2. Удаление связей с атрибутами.

Связи с атрибутами должны быть преобразованы в сущности.

В разработанной концептуальной модели существовала связь N:M («Раздельные пункты» : «Пути»), которая имела собственные атрибуты. После устранения связь N:M, ее атрибуты перешли в сущность «Раздельные пункты на пути» (см. рис. 7.9, неключевые атрибуты).

В графической нотации IDEF1X не предусмотрено отображение связей с атрибутами, хотя некоторые методологии ERD допускают их наличие и отображение.

1.3. Удаление сложных связей (со степенью участия более 2).

Сложную связь заменяют необходимым количеством бинарных связей 1:N со вновь созданной сущностью, которая и показывает эту связь.

1.4. Удаление рекурсивных связей (со степенью участия 1).

2. Проверка модели с помощью правил нормализации.

Основная идея нормализации заключается в том, чтобы каждый факт хранился в одном месте, т. е. чтобы не было дублирования данных. Многие из требований нормализации, как правило, уже учитываются при выполнении предыдущих шагов проектирования.

Ниже приводятся краткие сведения из теории нормализации.

Проектирование реляционной БД представляет собой пошаговый процесс создания набора отношений (таблиц, сущностей), в которых отсутствуют нежелательные функциональные зависимости.

Функциональная зависимость определяется следующим образом. Пусть A и B – произвольные наборы атрибутов отношения. Тогда B функционально зависит от A (A → B), в том и только в том случае, если каждому значению A соответствует в точности одно значение B. Левая часть функциональной зависимости (A) называется *детерминантом*, а правая (B) – *зависимой частью*. В частности, в отношении A может быть первичным ключом, а B – набором неключевых атрибутов, так как одному значению первичного ключа в точности соответствует одно значение набора неключевых атрибутов.

3.4. Физическое проектирование с использованием методологии IDEF1X

Цель физического проектирования – преобразование логической модели с учетом синтаксиса, семантики и возможностей выбранной целевой СУБД.

В связи с тем, что методология физического проектирования существенно зависит от выбранной целевой СУБД, ограничимся лишь общими рекомендациями [20, 21].

1. Анализ необходимости введения контролируемой избыточности.

При реализации проекта часто для достижения большей эффективности системы требуется снизить требования к уровню нормализации отношений, т. е. внести некоторую избыточность данных. Процесс внесения таких изменений в БД называется *денормализацией*.

Рассмотрим некоторые виды денормализации, которые в определенных случаях могут существенно повысить производительность системы.

1.1. Использование производных данных.

С точки зрения физического проектирования любой производный атрибут либо может сохраняться в БД, либо при каждом обращении к нему его значение будет вычисляться заново. Например, длина пути может каждый раз вычисляться по таблицам «Действительный километраж» и «Неправильные пикеты» либо храниться как атрибут в таблице «Пути».

Проектировщик при использовании производных данных должен оценить:

· дополнительную стоимость хранения производных данных и поддержки согласованности с текущими значениями тех данных, на основании которых они вычисляются, т.е. минусы хранения производных данных;

· издержки на выполнение вычислений значений производных атрибутов при каждом обращении к ним – плюсы.

1.2. Дублирование атрибутов.

1.2.1. Объединение отношений, связанных 1:1.

Даже в тех случаях, когда связь между двумя сущностями необязательная, стоит подумать об их объединении, с учетом того, что часть полей в записях не будет заполняться. Руководствоваться в таких случаях надо из тех же соображений, что и при использовании производных данных. В частности, на рис. 7.6 и 7.7 показана иерархия наследования для сущности «Сотрудник». Если эту иерархию оставить без изменений, то в БД будет четыре сущности (Сотрудник, Постоянный сотрудник, Совместитель и Консультант), согласованность которых надо будет поддерживать. При выполнении запросов по сотрудникам надо будет оперировать четырьмя таблицами. *1.2.2. Дублирование атрибутов в связях типа 1:M.*

Например, при запросе к таблице «Раздельные пункты на пути» очень часто будет требоваться наименование самих раздельных пунктов. С целью уменьшения нагрузки на БД следует рассмотреть возможность включения атрибута в эту таблицу.

1.2.3. Использование служебных таблиц (справочных таблиц, классификаторов, типовых списков значений).

Служебные таблицы, как правило, создаются для атрибутов символьного типа, значения которых могут выбираться из строго определенного и ограниченного списка. Например, значениями атрибута «Род балласта» могут быть только «Щебеночный», «Песчаный», «Гравийный» и «Асбестовый».

Раздел 4. Основы объектно-ориентированного подхода к проектированию и разработке программ
(комп. презентация – 6 час)

Терминология объектно-ориентированного программирования

Перед тем, как перейти к описанию преимуществ, которые дает ООП разработчикам программного обеспечения в процессе проектирования, кодирования и тестирования программных продуктов необходимо познакомиться с наиболее часто встречающимися терминами в этом области.

Класс – тип данных, описывающий структуру и поведение объектов.

Объект – экземпляр класса.

Поле – элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса.

Состояние объекта – набор текущих значений полей объекта.

Метод – процедура или функция, выполняющаяся в контексте объекта, для которого она вызывается. Методы могут изменять состояние текущего объекта или состояния объектов, передаваемых им в качества параметров.

Свойство – специальный вид методов, предназначенный для модификации отдельных полей объекта. Имена свойств обычно совпадают с именами соответствующих полей. Внешне работа со свойствами выглядит точно так же, как работа с полями структуры или класса, но на самом деле перед тем, как вернуть или присвоить новое значение полю может быть выполнен программный код, осуществляющий разного рода проверки, к примеру, проверку на допустимость нового значения.

Член класса – поля, методы и свойства класса.

Модификатор доступа – дополнительная характеристика членов класса, определяющая, имеется ли к ним доступ из внешней программы, или же они используются исключительно в границах класса и скрыты от окружающего мира. Модификаторы доступа разделяют все элементы класса на детали реализации и открытый или частично открытый интерфейс.

Конструктор – специальный метод, выполняемый сразу же после создания экземпляра класса. Конструктор инициализирует поля объекта – приводит объект в начальное состояние. Конструкторы могут быть как с параметрами, так и без. Конструктор без параметров называют конструктором по умолчанию, который может быть только один. Имя метода конструктора, чаще всего, совпадает с именем самого класса.

Деструктор – специальный метод, вызываемый средой исполнения программы в момент, когда объект удаляется из оперативной памяти. Деструктор используется в тех случаях, когда в состав класса входят ресурсы, требующие явного освобождения (файлы, соединения с базами данных, сетевые соединения и т.п.)

Интерфейс – набор методов и свойств объекта, находящихся в открытом доступе и призванных решать определенный круг задач, к примеру, интерфейс формирования графического представления объекта на экране или интерфейс сохранения состояния объекта в файле или базе данных.

Статический член – любой элемент класса, который может быть использован без создания соответствующего объекта. К примеру, если метод класса не использует ни одного поля, а работает исключительно с переданными ему параметрами, то ничто не мешает его использовать в контексте всего класса, не создавая отдельных его экземпляров. Константы в контексте класса обычно всегда являются статическими его членами.

- **Инкапсуляция** обозначает сокрытие деталей реализации классов средствами награждения отдельных его членов соответствующими модификаторами доступа. Таким образом, вся функциональность объекта, нацеленная на взаимодействие с другими объектами программы группируется в открытый интерфейс, а детали тщательно скрываются внутри, что избавляет основной код бизнес-логики информационной системы от ненужных ему вещей. Инкапсуляция повышает надежность работы программного кода, поскольку гарантирует, что определенные данные не могут быть изменены за пределами содержащего их класса.

- **Наследование**. Краеугольный камень ООП. В объектно-ориентированном программировании есть возможность наследовать структуру и поведение класса от другого класса. Класс, от которого наследуют, называется базовым или суперклассом, а класс, который получается вследствие наследования – производным или просто потомком. Любой класс может выступать как в роли суперкласса, так и в роли потомка. Связи наследования классов образуют иерархию классов. Множественным наследованием называют определение производного класса сразу от нескольких суперклассов. Не все объектно-ориентированные языки программирования поддерживают множественное наследование

- **Абстрагирование**. Возможность объединять классы в отдельные группы, выделяя общие, значимые для них всех характеристики (общие поля и общее поведение). Собственно, абстрагирование и есть следствие наследования: базовые классы не всегда имеют свою проекцию на объекты реального мира, а создаются исключительно с целью выделить общие черты целой группы объектов. К примеру, объект мебель – это базовое понятие для стола, стула и дивана, всех их объединяет то, что это движимое имущество, часть интерьера помещений, и они могут быть выполнены для дома или офиса, а также относиться к “эконом” или “премиум” классу. В ООП есть для этого отдельное понятие абстрактный класс – класс, объекты которого создавать запрещено, но можно использовать в качестве базового класса. Пример диаграммы классов, построенной путем абстрагирования, в ходе анализа видов существующих транспортных средств приведен на следующем рисунке. На верхних уровнях *иерархии наследования* находятся абстрактные классы, объединяющие транспортные средства по наиболее значимым характеристикам.

- **Полиморфизм**. Еще одно свойство, которое является следствием наследования. Дело в том, что объектно-ориентированные языки программирования позволяют работать с набором объектов из одной иерархии точно так же, как если бы все они были объектами их базового класса. Если вернуться к примеру про мебель, то можно предположить, что в контексте создания информационной системы для мебельного магазина в базовый класс для всех видов мебели разумно добавить общий для всех метод “показать характеристики”. При распечатке характеристик всех видов товара программа бы без разбору для всех объектов вызывала бы этот метод, а каждый конкретный объект уже сам бы решал, какую информацию ему предоставлять. Как это реализуется: Во-первых, в базовом классе определяют общий для всех метод с общим для всех поведением. В случае с нашим примером это будет метод, печатающий общие для любых типов мебели параметры. Во-вторых, в каждом производном классе, где это необходимо, переопределяют базовый метод (добавляют метод с тем же именем), где расширяют базовое поведение своим, например, выводят характеристики, свойственные только конкретному виду мебельной продукции. Метод в базовом классе иногда вообще не обязан содержать какой-либо код, а необходим только для того, чтобы определить имя и набор параметров – сигнатуру метода.

- **Интерфейс**. В некоторых языках программирования (C#, Java) понятие интерфейса выделено явно - это не только открытые методы и свойства самого класса. Такие языки, как правило, не поддерживают множественного наследования и компенсируют это тем, что любой объект может иметь один базовый объект и реализовывать любое количество интерфейсов. Интерфейс в их интерпретации – это подобие абстрактного класса, содержащего только описание (сигнатуру) открытых методов и свойств. Реализация интерфейса ложится на плечи каждого класса, который собирается его поддерживать. Один и тот же интерфейс могут реализовывать классы абсолютно разных иерархий, что расширяет возможности полиморфизма.

- Объектно-ориентированное программирование постоянно развивается, порождая новые парадигмы, такие как аспектно-ориентированное, субъектно-ориентированное и даже агентно-ориентиванное программирование. Нужно отметит, что лавры ООП не дают покоя остальным теоретикам, и они спешат предложить свои варианты его совершенствования и расширения. Про аспектно-ориентированное программирование я написал отдельную заметку, а сейчас хочу пару слов сказать про прототипное программирование, которое реализует язык веб-программирования на стороне клиента JavaScript. Прототипное программирование исключает понятие класса, заменяя его прототипом – образцом объекта. Таким образом, в прототипно-ориентированном языке нет понятия типа объекта, а есть понятие образец или прототип. **Прототип** – это экземпляр объекта, по которому создаются другие экземпляры, копируя (клонируя) его члены. В JavaScript вы не описываете поля и методы класса, а создаете сначала пустой объект, а потом добавляете ему нужные поля и методы (в JavaScript метод можно определить и добавить к объекту динамически). Точно также создаются и прототипы, на которые потом ссылаются другие объекты, как на свой прообраз. Если у объекта не находится какого-то метода или поля, которое указано в месте вызовы, то оно ищется среди членов его прототипа. То, как реализуется объектно-ориентированный подход в JavaScript, я также отдельно описал.

Некоторые элементы современного объектно-ориентированного программирования

Время не стоит на месте, да и времени с момента появления ООП уже прошло довольно много, поэтому не стоит удивляться, что сегодня словарь по объектно-ориентированному программированию серьезно разросся. Итак, вот некоторые новые термины и понятия, связанные с ООП.

- **События**. Специальный вид объектов, создаваемый для оповещения одних объектов о событиях, происходящих с другими объектами. В разных языках программирования механизм событий реализуется по-разному: где-то с помощью специальных синтаксических конструкции, а где-то силами базовых средств ООП.

- **Универсальный тип**. Концепция универсальных типов не связана непосредственно с концепцией ООП, но она является причиной появление таких элементов, как универсальный класс, универсальный метод, универсальное событие и т.д. Универсальный тип – это тип, параметризованный другим типом (набором типов). Кем является этот тип-параметр в контексте проектирования универсального типа неизвестно, хотя есть возможность ограничить значения типов-параметров, заставив их быть производными от конкретного класса или реализовывать определенные интерфейсы.

- **Исключения**. Еще один специальный вид объектов, поддерживаемый встроенным в конкретный язык программирования механизмом обработки ошибок и исключительных ситуаций. Исключения, помимо кода ошибки, содержат ее описание, возможные причины возникновения и стек вызовов методов, имевший место до момента возникновения исключения в программе.

Недостатки объектно-ориентированного программирования

Про то, что популярность объектно-ориентированного подхода к созданию программных продуктов огромна я уже сказал. Про то, что тех, кто стремится расширить эту парадигму довольно много, я тоже уже отметил. Но есть еще один способ выделиться среди огромного сообщества специалистов в информационных технологиях – это заявить, что ООП себя не оправдало, что это не панацея, а, скорее, плацебо. Есть среди этих людей действительно специалисты очень высокого класса, такие как Кристофер Дэйт, Александр Степанов, Эдсгер Дейкстра и другие, и их мнение заслуживает внимания, но есть и те, про которых говорят, что “плохому танцору всегда что-то мешает”. Вот они, наиболее очевидные недостатки ООП, на которые указывают специалисты:

1. ООП порождает огромные иерархии классов, что приводит к тому, что функциональность расползается или, как говорят, размывается по базовым и производным членам класса, и отследить логику работы того или иного метода становится сложно.

2. В некоторых языках все данные являются объектами, в том числе и элементарные типы, а это не может не приводить к дополнительным расходам памяти и процессорного времени.

3. Также, на скорости выполнения программ может неблагоприятно сказаться реализация полиморфизма, которая основана на механизмах позднего связывания вызова метода с конкретной его реализацией в одном из производных классов.

4. Психологический аспект. Многие считают, что ООП это круто и начинают использовать его подходы всегда и везде и без разбору. Все это приводит к снижению производительности программ в частности и дискредитации ООП в целом. Ну, про плохих танцоров я уже сказал…

Основы объектно-ориентированного подхода

Выберем предметную область, объекты которой будем описывать средствами языка C#. Пусть это будет что-то из области геоинформационных систем (ГИС), тема которых близка автору этого материала. Начнем с малого и опишем класс *ориентированной точечной геометрии*, определяемой точкой и вектором (ориентацией геометрии) на плоскости. На данный момент нет желания углубляться в детали механизмов графического представления пространственных данных в ГИС, но стоит отметить, что с использованием точечной геометрии отображают объекты, размеры которых и протяженность на выбранном масштабе карты значения не имеют, а важно только их местоположение. Точечную геометрию могут отображать на картах ГИС не только в виде круглой точки, но и в виде произвольного символа, например, стрелки. Местоположение стрелки определяют координаты X и Y, а ее направление как раз и задается той самой ориентацией точечной геометрии. В ГИС обычно точки и векторы имеют три координаты, но здесь мы не будем усложнять картину мира и остановимся на двух. Поскольку понятие точки и вектора нам могут понадобиться не только для описания точечной геометрии, то имеет смысл их структуру определить в отдельных от ориентированной точечной геометрии классах. Примеры определения классов точки и вектора с подробными комментариями приведены ниже.

Класс Point2D имеет два конструктора: один без параметров или *конструктор по умолчанию*, и второй – конструктор с параметрами.

Конструктор - специальная процедура, которая запускается в момент создания нового экземпляра класса (объекта).

Конструктор "конструирует" объект, инициализируя его поля конкретными значениями.

Определение конструктора с параметрами содержит вызов конструктора без параметров. Хотя здесь в этом смысла особого и нет, но на практике такая возможность крайне полезна, поскольку позволяет выделить общую часть инициализации класса в отдельном конструкторе. Точно также в любой конструктор можно поместить вызов любого другого конструктора (не только конструктора по умолчанию, которого может и не быть вовсе) с любым модификатором доступа: *private, protected, internal* и т.д.. Естественно, что в первую очередь выполняется код указанного после двоеточия конструктора, а потом уже код тела вызывающего конструктора.

Если вы хотите запретить создание ваших объектов посредством оператора *new* извне, то определите конструктор по умолчанию с модификатором *private* и создавайте экземпляры класса с помощью метода фабрики – любого статического метода вашего класса. Если вы хотите, чтобы конструктор по умолчанию был запрещен только пользователям вашего SDK

Software Development Kit (SDK) - набор программных компонентов, на основе которых создаются приложения.

, то сделайте его *internal, protected* или *internal protected* в зависимости от ваших целей. Модификатор *internal* может подойти для случая, когда экземпляры вашего класса создаются с помощью методов *класса фабрики*, определение которой находится в одной с вашим классом *сборке* (проекте). Такой прием следует применять, если требуется скрыть от внешнего мира некоторые элементы окружения вашей программы, которые вы используете для инициализации экземпляров класса, например: идентификатор активного соединения с базой данных. Модификатор *protected* подходит для случаев, когда конструктор играет вспомогательную роль: частично выполняет инициализацию класса - общую часть для всех производных классов, завершить которую в полном объеме может только экземпляр производного класса. Такой подход используют при определении абстрактных классов, которые будут рассмотрены в следующих разделах.

4.3. Лабораторные работы

№ п/п	*Номер раздела дисциплины*	*Наименование тем лабораторных работ*	*Объем (час.)*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1.	**1.**	Разработка функциональной модели (методология IDEF0)	4	-
2.	**2.**	Разработка функциональной модели (методология DFD)	2	-
3.	**3.**	Проектирование реляционной базы данных как компонента АИС	4	-
4.	**4.**	Объектно-ориентированное проектирование пользовательского интерфейса	8	работа в малых группах (8 час.)
		ИТОГО	**18**	**8**

4.4. Практические занятия

Учебным планом не предусмотрено

4.5. Курсовая работа

Тема. Разработка пользовательского интерфейса реляционной базы данных на основе механизма BDE.

Цель работы. Приобретение и закрепление практических навыков функционального подхода при решении задач информационного менеджмента.

Курсовая работа представляет собой самостоятельно выполненное обучающимся логически завершенное исследование. При выполнении курсовой работы обучающийся должен:

- совершенствовать теоретические знания по дисциплине «Разработка программных приложений»;
- продемонстрировать способность обобщать, систематизировать и анализировать информацию, необходимую для проведения исследования и решения поставленных задач;
- приобрести практические навыки в освоении методологии разработки программных приложений.

Для выполнения курсовой работы преподаватель предоставляет обучающемуся тему курсовой работы в соответствии с номером варианта индивидуально.

Результаты выполнения курсовой работы оформляются в виде пояснительной записки объемом 40-45 листов в строгом соответствии со стандартом ФГБОУ ВО «БрГУ».

5. МАТРИЦА СООТНЕСЕНИЯ РАЗДЕЛОВ УЧЕБНОЙ ДИСЦИПЛИНЫ К ФОРМИРУЕМЫМ В НИХ КОМПЕТЕНЦИЯМ И ОЦЕНКЕ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Компетенции / *№, наименование разделов дисциплины*	*Кол-во часов*	*Компетенции*			*Σ комп.*	*$t_{ср}$, час*	*Вид учебной работы*	*Оценка результатов*
		ПК						
		2	*7*	*8*				
1	2	3	4	5	6	7	8	
1. Основы разработки программных приложений	**24**	+	+	+	3	8	Лк, ЛР, СРС	КР, экзамен
2. Технологии и подходы к проектированию программных приложений	**21**	-	+	-	1	21	Лк, ЛР, СРС	КР, экзамен
3. Разработка информационной модели	**25**	-	+-	-	1	25	Лк, ЛР, СРС	КР, экзамен
4. Основы объектно-ориентированного подхода к проектированию и разработке программ	**38**	-	-	+	1	38	Лк, ЛР, СРС	КР, экзамен
всего часов	**108**	**8**	**54**	**46**	**3**	**36**		

6. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Шичкина Ю.А. Разработка приложений для работы с базами данных в среде программирования VisualStudio C#. В 2 ч. Ч.1,2 / Ю. А. Шичкина, В. С. Кедрин . - Братск : БрГУ, 2013.Ч.1 : Разработка программных приложений на базе SQL serverCompact 3.5. - 2013. - 100 с.
2. Шичкина Ю.А. Разработка приложений для работы с базами данных в среде программирования VisualStudio C#. В 2 ч. Ч.1,2 / Ю. А. Шичкина. - Братск :БрГУ, 2013. Ч.2 : Клиент-серверные и XML-ориентированные Разработка программных приложений с доступом к данным ASP.NET. - 156 с.
3. Ашарина И.В. Язык С ++ и объектно-ориентированное программирование в С ++. Лабораторный практикум : учебное пособие / И. В. Ашарина, Ж. Ф. Крупская . - Москва : Горячая линия- Телеком, 2015. - 232 с.
4. Затонский А.В. Информационные технологии: разработка информационных моделей и систем : учебное пособие / А. В. Затонский. - Москва : РИОР; ИНФРА-М, 2014. - 344 с. - (Высшее образование: Бакалавриат).
5. Заботина Н.Н. Проектирование информационных систем : учебное пособие / Н. Н. Заботина. - М. : ИНФРА-М, 2013. - 331 с. - (Высшее образование: Бакалавриат).
6. Губарева Т.В. Программные средства разработки WEB-страниц : учебное пособие / Т. В. Губарева. - Братск : БрГУ, 2009. - 302 с
7. Орлов С.А. Технологии разработки программного обеспечения. Современный курс по программной инженерии : учебник для вузов / С. А. Орлов, Б. Я. Цилькер. - 4-е изд. - Санкт-Петербург : Питер, 2012. - 608 с. - (Учебник для вузов. Стандарт третьего поколения).

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

№	*Наименование издания*	*Вид занятия* (	*Количество экземпляров в библиотеке,* **шт.**	*Обеспеченность,* **(экз./ чел.)**
1	**2**	**3**	**4**	**5**
	Основная литература			
1	Бикмухаметов, И.Х. Разработка учетных приложений в среде MS Office : учебное пособие / И.Х. Бикмухаметов, З.Ф. Исхаков, М.Ю. Лехмус ; Финансовый университет при Правительстве РФ. - Москва : Прометей, 2018. - 121 с. : ил. - Библиогр. в кн. - ISBN 978-5-907003-16-3 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=494922	Лк, ЛР, КР, СР	1(ЭР)	1

2	Зубкова, Т.М. Технология разработки программного обеспечения : учебное пособие / Т.М. Зубкова ; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего образования «Оренбургский государственный университет», Кафедра программного обеспечения вычислительной техники и автоматизированных систем. - Оренбург : ОГУ, 2017. - 469 с. : ил. - Библиогр.: с. 454-459 - ISBN 978-5-7410-1785-2 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=485553	Лк, ЛР, КР, СР	1(ЭР)	1
3	Илюшечкин В.М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин . - Москва : Юрайт, 2016. - 213 с. - ISBN 978-5-9916-4705-2	Лк, ЛР, КР, СР	10	0,6
4	Подбельский, В. В.Язык C#. Базовый курс [Электронный ресурс] : учебное пособие / В. В. Подбельский. - 2-е изд., перераб. и доп. - Москва : Финансы и статистика, 2015. - 408 с. http://ecat.brstu.ru/catalog/Ресурсы%20свободного%20доступа/Подбельский%20В.В.Язык%20Си.Базовый%20курс.Учеб.пособие.2015.pdf	Лк, ЛР, КР, СР	1(ЭР)	1
	Дополнительная литература			
5	Дьяконица С. А. Основы программирования на языке Си/Си ++ : лабораторный практикум / С. А. Дьяконица, Д. С. Семенов. - Братск : БрГУ, 2015. - 153 с. -	ЛР, КР, СР	45	1
6	Гагарина Л.Г. Технология разработки программного обеспечения : учебное пособие для вузов / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Виснадул. - Москва : ИНФРА-М, 2009. - 400 с. - (Высшее образование).	Лк, ЛР, КР, СР	15	1
7	Шичкина Ю.А. Разработка приложений для работы с базами данных в среде программирования Visual Studio C#. В 2 ч. Ч.1,2 / Ю. А. Шичкина, В. С. Кедрин . - Братск : БрГУ, 2013. Ч.1 : Базы данных на базе SQL server Compact 3.5. - 2013. - 100 с.	ЛР, КР, СР	90	1
8	Шичкина Ю.А. Разработка приложений для работы с базами данных в среде программирования Visual Studio C#. В 2 ч. Ч.1,2 / Ю. А. Шичкина. - Братск : БрГУ, 2013. Ч.2 : Клиент-серверные и XML-ориентированные базы данных с доступом к данным ASP.NET.- 156 с.	ЛР, КР, СР	90	1
9	Благодатских, В.А. Стандартизация разработки программных средств : учебное пособие для вузов / В.А. Благодатских, В.А. Волнин, К.Ф. Поскакалов. - Москва : Финансы и статистика, 2006. - 284 с.	Лк, ЛР, КР, СР	30	1

8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО - ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ» НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1. Электронный каталог библиотеки БрГУ: http://irbis.brstu.ru/CGI/irbis64r_15/cgiirbis_64.exe?LNG=&C21COM=F&I21DBN=BOOK&P21DBN=BOOK&S21CNR=&Z21ID=
2. Электронная библиотека БрГУ http://ecat.brstu.ru/catalog
3. Федеральная университетская компьютерная сеть России // Электронный ресурс [Режим

доступа: свободный] http://www.runnet.ru/

4. Каталог учебников, оборудования, электронных ресурсов // Электронный ресурс [Режим доступа: свободный] http://ndce.edu.ru/

5. Электронно-библиотечная система издательства «Лань» // Электронный ресурс http://e.lanbook.com/ ,

6. Библиотека «Книгосайт» // Электронный ресурс [Режим доступа: свободный] http://knigosite.ru/

7. Электронная библиотека книг на тему бизнеса, финансов, экономики и смежным темам // Электронный ресурс [Режим доступа: свободный] http://www.finbook.biz/

8. ЭБС «Университетская библиотека online» // Электронный ресурс http://biblioclub.ru/ ,

9. Научная электронная библиотека «КИБЕРЛЕНИНКА» // Электронный ресурс [Режим доступа: свободный] http://cyberleninka.ru/

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Вид учебных занятий	Организация деятельности обучающихся
Лекции	Написание конспекта лекций: кратко, последовательно фиксировать основные положения, выводы, формулировки, обобщения; помечать важные мысли, выделять ключевые слова, термины. Проверка терминов с помощью энциклопедий, словарей, справочников с выписыванием толкований в тетрадь. Обозначить вопросы, термины, материал, который вызывает трудности, пометить и попытаться найти ответ в рекомендуемой литературе. Если самостоятельно не удается разобраться в материале, необходимо сформулировать вопрос и задать преподавателю на консультации, практическом занятии.
Лабораторные занятия	Развитие интеллектуальных умений, практических навыков, подготовка ответов к контрольным вопросам, работа с основной и дополнительной литературой, необходимой для освоения дисциплины, выполнение заданий, активное участие в интерактивной, активной, инновационной формах обучения, составление и оформление отчетов по лабораторным работам.
Курсовая работа	Работа с основной и дополнительной литературой, необходимой для выполнения курсовой работы, углубление и конкретизация необходимого в соответствии с темой материала из литературных источников и полученных теоретических знаний, выработка способности и готовности их использования в практической исследовательской работе. Развитие интеллектуальных умений изложения материала, представления с элементами визуализации (схемы, графики, рисунки, таблицы, формулы) и оформления в соответствии с требованиями ГОСТ.
Самостоятельная работа обучающихся	*Подготовка к лабораторным занятиям.* Проработка основной и дополнительной литературы, терминов, сведений, требующихся для запоминания и являющихся основополагающими в теме/разделе. Конспектирование прочитанных литературных источников. Проработка материалов по изучаемому вопросу, с использованием на рекомендуемых ресурсов информационно-телекоммуникационной сети «Интернет». Выполнение заданий преподавателя, необходимых для подготовки к участию в интерактивной, активной, инновационных формах обучения по изучаемой теме. *Подготовка к экзамену.* При подготовке к экзамену необходимо ориентироваться на конспекты лекций, рекомендуемую литературу, использовать рекомендуемые ресурсы информационно-телекоммуникационной сети «Интернет».

9.1. Методические указания для обучающихся по выполнению лабораторных работ

Лабораторная работа № 1. Разработка функциональной модели (методология IDEF0)

Цель лабораторной работы. Освоение CASE-средства BPwin в целях разработки функциональной модели информационной системы с использованием методологии IDEF0.

Задание на выполнение лабораторной работы

1) Изучить и закрепить основы разработки функциональных моделей с использованием методологии IDEF0 (Лекция. Разработка функциональной модели).

2) Освоить CASE-средство BPwin в части разработки функциональных моделей с использованием методологии IDEF0.

3) Построить функциональную модель информационной системы по индивидуальному заданию.

4) Оформить и защитить отчет. В отчете должны быть приведены:

- контекстная диаграмма;
- диаграмма декомпозиции 1-го уровня;
- две диаграммы декомпозиции 2-го уровня для двух наиболее интересных блоков с диаграммы декомпозиции 1-го уровня;
- диаграмма дерева узлов.

Назначение BPwin. CASE-средство BPwin предназначено для построения функциональных моделей с использованием методологий:

- IDEF0 - функциональные модели любых систем;
- IDEF3 - функциональные модели технологических процессов;
- DFD - функциональные модели информационных систем.

Навигатор панели процессов предназначен для отображения и выбора диаграмм разрабатываемой функциональной модели.

Рабочая область предназначена для отображения и редактирования диаграммы модели, выбранной в панели процессов.

Создание новой модели

Для создания новой модели необходимо выбрать пункт меню "File / New" или нажать на соответствующую кнопку стандартной панели инструментов На экране появится диалоговое окно

В диалоговом окне необходимо выбрать радиокнопку "Create model", ввести имя модели в поле "Name" и выбрать методологию, нотация которой будет использовать при построении модели (радиокнопки "Type").

Для указания общих параметры модели необходимо выбрать пункт меню "Model / Model Properties" и в появившемся диалоговом окне перейти на вкладку "General

На вкладке задаются следующие параметры модели:

- имя модели (Model name);
- имя проекта (Project). Имя проекта, как правило, совпадает с именем разрабатываемой информационной системы;
- фамилия автора или наименование компании (Author);
- инициалы автора (Author initials);
- тип модели - AS-IS (как есть) или TO-BE (как будет). Подробнее см. раздел "Основы функционального анализа и проектирования систем".

Создание и редактирование диаграмм

После нажатия на кнопку "Ok" диалогового окна создания модели автоматически создается контекстная диаграмма. Указание параметров диаграммы, выбранной в текущий момент в панели процессов, осуществляется через диалоговое окно "Diagram Property", вызываемого через пункт меню "Diagram / Diagram Property"

На вкладке "Status" указываются статус, дата создания и дата последней редакции диаграммы

Типы статуса диаграммы имеют следующий смысл:

- рабочая (WORKING) – диаграмма находится в стадии разработки;
- черновик (DRAFT) – диаграмма прошла некоторые стадии рассмотрения с заказчиками, но это не окончательный вариант;
- рекомендованная (RECOMMENDED) – диаграмма прошла все стадии рассмотрения с заказчиками и отвечает формальным требованиям, но это не окончательный вариант;
- готовая или публикуемая (PUBLICATION) – окончательный вариант диаграммы.

На вкладке "Page Setup" указываются единицы измерения (Units), формат листов (Sheet Size), поля, необходимость отображения заголовка (Header) и нижнего колонтитула (Footer

На вкладке "Header/Footer" возможно задание пользовательского (custom) вида заголовка (Header) и нижнего колонтитула (Footer) диаграммы

а вкладке диалогового окна можно задать:

- имя блока (вкладка "Name");
- комментарий к блоку (вкладка "Definition");
- параметры шрифта надписи блока (вкладка "Font");
- цвет блока (вкладка "Color");
- графический примитив, используемый для отображения блока (вкладка "Box style").

Контрольные вопросы для самопроверки

1. Case-технологии анализа и проектирования ИС. Назначение и основные возможности Case-средств.
2. Модели ИС.
3. Принципы построения моделей.
4. Сущность структурного подхода к анализу и проектированию ИС.
5. Краткая характеристика методологий структурного анализа и проектирования ИС.

6. Методология IDEF0. Назначение.
7. Методология IDEF0. Виды диаграмм.
8. Методология IDEF0. Элементы диаграмм.
9. Методология IDEF0. ICOM-коды.
10. Методология IDEF0. Типы связей между работами.
11. Методология IDEF0. Модели AS-IS, TO-BE и SHOULD-BE.
12. Методология IDEF0. Правила и рекомендации построения диаграмм.

Основная литература

[1.- 2.] из раздела 7.

Дополнительная литература

[3.- 7.] из раздела 7.

Лабораторная работа № 2. Разработка функциональной модели (методология DFD)

Цель лабораторной работы: Освоение CASE-средства BPwin в целях разработки функциональной модели информационной системы с использованием методологии DFD.

Задание на выполнение лабораторной работы

1) Изучить и закрепить основы разработки функциональных моделей с использованием методологии DFD (материалы лекции 6. Разработка функциональной модели).

2) Освоить CASE-средство BPwin в части разработки функциональных моделей с использованием методологии DFD.

3) Построить функциональную модель информационной системы по индивидуальному заданию.

4) Оформить и защитить отчет. В отчете должны быть приведены:

- контекстная диаграмма;
- диаграмма декомпозиции 1-го уровня;
- две диаграммы декомпозиции 2-го уровня для двух наиболее интересных блоков с диаграммы декомпозиции 1-го уровня;
- диаграмма дерева узлов.

На диаграммах должны быть показаны не менее 3-4 внешних сущностей и 4-5 накопителей данных.

Назначение, общие сведения об интерфейсе, создание новой модели и основы создания и редактирования диаграмм рассмотрены в предыдущем занятии (№ 1. Разработка функциональной модели (методология IDEF0)).

Ниже рассматриваются особенности разработки функциональной модели с использованием методологии DFD:

Панель инструментов "BPwin Toolbox" при редактировании диаграмм в нотации DFD имеет следующий вид Для указания параметров процесса, потока данных, внешней сущности и накопителя данных используются диалоговые окно, аналогичные окнам "Activity Properties" и "Arrow Properties".

Задание особых видов потоков данных (двунаправленного, квазинепрерывного и управляющего) осуществляется в разделе "Type" на вкладке "Style" диалогового окна "Arrow Properties"

Контрольные вопросы для самопроверки

1. Диаграммы потоков данных. Назначение.
2. Диаграммы потоков данных. Элементы диаграмм.
3. Диаграммы потоков данных. Правила и рекомендации построения диаграмм. Миниспецификации.
4. Диаграммы потоков данных. Расширение DFD для систем реального времени.
5. ERD. Назначение и основные элементы моделей.
6. Возможности современных CASE-средств моделирования данных.

Основная литература

1.- 2. из раздела 7.

Дополнительная литература

3.- 7. из раздела 7.

Лабораторная работа № 3. Проектирование реляционной базы данных как компонента АИС

Цель лабораторной работы

Освоение CASE-средства ERwin в целях разработки информационной модели с использованием методологии IDEF1X.

Задание на выполнение лабораторной работы

1) Изучить и закрепить основы разработки информационных моделей с использованием методологии IDEF1X (материалы лекции 7. Разработка информационной модели).

2) Освоить CASE-средство ERwin в части разработки информационных моделей с использованием методологии IDEF1X .

3) Построить информационную модель системы по индивидуальному заданию.

4) Оформить и защитить отчет. В отчете должны быть приведены:

- концептуальная модель БД;
- логическая модель БД;
- физическая модель БД, включая типы данных для атрибутов и триггеры;
- DDL-скрипт генерации структуры БД.

CASE-средство ERwin предназначено для разработки информационных моделей с использованием методологий IDEF1X и IE. В ERwin реализованы основные функции, характерные для классических CASE-средств:

- прямое проектирование от создания концептуальной или логической модели БД до генерации **структуры БД** на диске или DDL-скрипта;
- обратное проектирование (реинжиниринг) создания физической модели БД на основе БД на диске или DDL-скрипта;
- синхронизацию моделей БД с самой БД на диске.

В качестве несомненных достоинств Erwin следует отметить:

- поддержку около 20 промышленных СУБД (ORACLE, Informix, DB2, MS SQL Server и др.) и 5 популярных настольных СУБД (Access, Foxpro, Paradox и др.);
- наличие функции проверки моделей БД требованиям полноты, целостности и нормализации.

Общие сведения об интерфейсе ERwin

Внешний вид главного окна ERwin представлен на рис.1.

Навигатор панели процессов предназначен для отображения и выбора общих доменов (Domains), правил проверки данных (Validation Rules), значений по умолчанию (Default Values), сущностей (Entities) и др., используемых при построении информационной модели.

Рабочая область предназначена для отображения и редактирования диаграммы модели.

Создание новой модели

Для создания новой модели необходимо выбрать пункт меню "File / New" или нажать на соответствующую кнопку стандартной панели инструментов (см.рис.2). На экране появится диалоговое окно В диалоговом окне необходимо выбрать тип модели "New Model Type" (логическая, физическая или логическая/физическая), а также выбрать целевую СУБД "Database" и ее версию "Version".

Для указания общих параметры модели необходимо выбрать пункт меню "Model / Model Properties" и в появившемся диалоговом окне перейти на вкладку "General" В разделе "Auto-Transform Logical Objects" имеется возможность указание следующих действий при переходе от логической модели к физической:

- преобразование связи многие-ко-многим в связующую таблицу, соединенную с исходными связями один-ко-многим (Many-to-Many Relationships with Association Table);
- автоматическое создание триггеров для суперклассов и подклассов (Supertype/Subtype with Identifying Relationships).

На вкладке "Notation" можно изменить нотацию, в соответствии с которой отображаются диаграммы (рис.5).

Создание и редактирование диаграмм

Перед созданием диаграммы необходимо в списке панели инструментов "Standard" выбрать ее тип: логическая или физическая.

Для создания элементов диаграммы используется панель инструментов "Toolbox" (отображение или скрытие панели выполняется через пункт меню "View"). На рис.6 приведено назначение элементов управления панель инструментов "Toolbox".

Для работы с сущностями предназначено контекстное меню, вызываемое при нажатии правой кнопки мыши на соответствующей сущности (рис.7).

Для указания общих параметров сущности используется диалоговое окно "Entities" (рис.8), вызываемое через пункт "Entity Properties" контекстного меню.

В диалоговом окне задаются имя и комментарий к сущности. В случае установки галочки напротив пункта "Logical Only" сущность будет отображаться только в логической модели.

Для указания набора атрибутов сущности и их параметров используется диалоговое окно "Attributies" (рис.9), вызываемое через пункт "Attributes" контекстного меню.

В левой части окна отображается список имеющихся атрибутов сущности с возможностями:

- добавления нового атрибута - кнопка "New";
- переименования выделенного атрибута - кнопка "Rename";
- удаления выделенного атрибута - кнопка "Delete";
- изменения положения выделенного атрибута в списке - кнопки "▲" и "▼".

При задании нового атрибута посредством нажатия кнопки "New" требуется указать его тип, имя атрибута в логической модели (Attribute Name) и имя атрибута в физической модели (Column Name) (рис.10).

В правой части диалогового окна "Attributies" для выделенного атрибута задается один из стандартных типов данных (на вкладке "General") и указывается признак вхождения атрибута в первичный ключ (Primary Key).

На вкладке "Datatype" диалогового окна "Attributies" задаются дополнительные параметры выделенного атрибута .

На вкладке можно задать более конкретный тип данных атрибута (Datatype), правило проверки данных при изменении значения атрибута (Valid) и значение по умолчанию (Default).

Для создания связи между сущностями необходимо в панели инструментов "Standard" нажать на кнопку соответствующей связи, щелкнуть левой кнопкой мыши вначале по родительской, а затем по дочерней сущности. Если в родительской сущности имеются атрибуты, входящие в первичный ключ, то эти атрибуты будут добавлены в дочернюю сущность в качестве внешнего ключа. При этом, если связь идентифицирующая, то добавленные атрибуты автоматически будут входить в состав первичного ключа дочерней сущности.

Для работы со связями предназначено контекстное меню, вызываемое при нажатии правой кнопки мыши на соответствующей связи .

Особенности работы с физической моделью

При работе с физической моделью в меню программы появляется пункт меню "Database". При выборе пунктов этого меню появляется возможность задания триггеров (Triggers), хранимых процедур (Stored Procedures) и SQL-скриптов (Pre & Post Scripts), выполняемых до и/или после создания БД или отдельной сущности. Триггеры, процедуры и скрипты могут быть заданы на уровне конкретных команд SQL.

Доступ к этим возможностям возможен также, но только на уровне сущности, через контекстное меню сущности, имеющее измененный вид

При выборе пункта "Columns" контекстного меню вместо вкладки "Datatype" появляется вкладка, соответствующая целевой СУБД На этой вкладке список типов данных ограничен набором типов, поддерживаемых целевой СУБД, и имеется возможность указание обязательности заполнения атрибута (NOT NULL).

Контрольные вопросы для самопроверки

1. Методология IDEF1X. Элементы диаграмм.
2. Концептуальное проектирование БД. Стадии.
3. Концептуальное проектирование БД. Сущности.
4. Концептуальное проектирование БД. Связи.
5. Концептуальное проектирование БД. Атрибуты.
6. Концептуальное проектирование БД. Ключи.
7. Концептуальное проектирование БД. Суперклассы и подклассы.
8. Логическое проектирование БД. Стадии.
9. Логическое проектирование БД. Удаление элементов, не отвечающих реляционной модели данных.
10. Логическое проектирование БД. Нормализация.
11. Логическое проектирование БД. Определение требований поддержки целостности данных.
12. Физическое проектирование БД. Стадии.
13. Физическое проектирование БД. Денормализация.
14. Физическое проектирование БД. Разработка механизмов защиты.

Основная литература

[1.- 4]. из раздела 6.

Дополнительная литература

[5.- 9]. из раздела 6.

Лабораторная работа № 4. Объектно-ориентированное проектирование пользовательского интерфейса *(работа в малых группах – 8 час)*

Цель лабораторной работы — изучение структуры и инструментов человекоориентированного подхода к разработке интерфейсов, основных принципов и паттернов разработки пользовательских интерфейсов, формирование навыков анализа, подбора и применения необходимого инструментария для решения поставленных интерфейсных задач.

Задание на выполнение лабораторной работы

1. Создать формы для ввода каждой из таблиц-справочников.
2. Создать сложную форму для таблиц, связанных отношением «1 ко многим».
3. Создать кнопочную форму, которая бы предоставляла доступ ко всем созданным формам и запросам.
4. Поместить в созданные формы кнопки навигации по записям и работы с формой (закрыть, напечатать, выйти из приложения и т.д.).
5. Создать модули для автоматической загрузки кнопочной формы при открытии базы данных.
6. Разработать руководство пользователя.

Отчет должен содержать:

– основную цель работы;
– описание последовательности выполнения задания;
– распечатки схем данных и создаваемых экранных форм и отчетов;
– распечатку разработанного руководства пользователя.

Терминология объектно-ориентированного программирования

Перед тем, как перейти к описанию преимуществ, которые дает ООП разработчикам программного обеспечения в процессе проектирования, кодирования и тестирования программных продуктов необходимо познакомиться с наиболее часто встречающимися терминами в этом области.

Класс – тип данных, описывающий структуру и поведение объектов.

Объект – экземпляр класса.

Поле – элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса.

Состояние объекта – набор текущих значений полей объекта.

Метод – процедура или функция, выполняющаяся в контексте объекта, для которого она вызывается. Методы могут изменять состояние текущего объекта или состояния объектов, передаваемых им в качества параметров.

Свойство – специальный вид методов, предназначенный для модификации отдельных полей объекта. Имена свойств обычно совпадают с именами соответствующих полей. Внешне работа со свойствами выглядит точно так же, как работа с полями структуры или класса, но на самом деле перед тем, как вернуть или

присвоить новое значение полю может быть выполнен программный код, осуществляющий разного рода проверки, к примеру, проверку на допустимость нового значения.

Член класса – поля, методы и свойства класса.

Модификатор доступа – дополнительная характеристика членов класса, определяющая, имеется ли к ним доступ из внешней программы, или же они используются исключительно в границах класса и скрыты от окружающего мира. Модификаторы доступа разделяют все элементы класса на детали реализации и открытый или частично открытый интерфейс.

Конструктор – специальный метод, выполняемый сразу же после создания экземпляра класса. Конструктор инициализирует поля объекта – приводит объект в начальное состояние. Конструкторы могут быть как с параметрами, так и без. Конструктор без параметров называют конструктором по умолчанию, который может быть только один. Имя метода конструктора, чаще всего, совпадает с именем самого класса.

На этом с терминологией ООП далеко еще не все, но остальные понятия, связанные с этой парадигмой будут рассмотрены в следующем разделе.

Недостатки объектно-ориентированного программирования

Про то, что популярность объектно-ориентированного подхода к созданию программных продуктов огромна я уже сказал. Про то, что тех, кто стремится расширить эту парадигму довольно много, я тоже уже отметил. Но есть еще один способ выделиться среди огромного сообщества специалистов в информационных технологиях – это заявить, что ООП себя не оправдало, что это не панацея, а, скорее, плацебо. Есть среди этих людей действительно специалисты очень высокого класса, такие как Кристофер Дэйт, Александр Степанов, Эдсгер Дейкстра и другие, и их мнение заслуживает внимания, но есть и те, про которых говорят, что “плохому танцору всегда что-то мешает”. Вот они, наиболее очевидные недостатки ООП, на которые указывают специалисты:

5. ООП порождает огромные иерархии классов, что приводит к тому, что функциональность расползается или, как говорят, размывается по базовым и производным членам класса, и отследить логику работы того или иного метода становится сложно.

6. В некоторых языках все данные являются объектами, в том числе и элементарные типы, а это не может не приводить к дополнительным расходам памяти и процессорного времени.

7. Также, на скорости выполнения программ может неблагоприятно сказаться реализация полиморфизма, которая основана на механизмах позднего связывания вызова метода с конкретной его реализацией в одном из производных классов.

8. Психологический аспект. Многие считают, что ООП это круто и начинают использовать его подходы всегда и везде и без разбору. Все это приводит к снижению производительности программ в частности и дискредитации ООП в целом. Ну, про плохих танцоров я уже сказал…

Контрольные вопросы для самопроверки

1. Класс. Объект. Состояние.
2. Метод Свойство Конструктор
3. Интерфейс
4. Полиморфизм
5. Преимущества объектно-ориентированного программирования
6. Недостатки объектно-ориентированного программирования.

Основная литература

[1.- 4]. из раздела 6.

Дополнительная литература

[5.- 9]. из раздела 6.

9.2. Методические указания по выполнению курсовой работы

Порядок выполнения курсовой работы.

Тема. Разработка пользовательского интерфейса реляционной базы данных на основе механизма BDE.

Цель работы. Приобретение и закрепление практических навыков функционального подхода при решении задач информационного менеджмента.

Курсовая работа представляет собой самостоятельно выполненное обучающимся логически завершенное исследование. При выполнении курсовой работы обучающийся должен:

- совершенствовать теоретические знания по дисциплине «Разработка программных приложений»;
- продемонстрировать способность обобщать, систематизировать и анализировать информацию, необходимую для проведения исследования и решения поставленных задач;
- приобрести практические навыки в освоении методологии разработки программных приложений.

Для выполнения курсовой работы преподаватель предоставляет обучающемуся в соответствии с номером варианта индивидуально.

Результаты выполнения курсовой работы оформляются в виде пояснительной записки объемом 40-45 листов.

Структурные элементы пояснительной записки:

- титульный лист;
- содержание;
- введение;
- основная часть,
- заключение;
- список использованных источников;
- приложения.

Содержание включает последовательно перечисленные наименования всех разделов, подразделов и приложений с указанием номеров страниц.

Во введении формулируются актуальность, цель и задачи курсовой работы, а также указываются методы и подходы для решения реализации поставленной цели.

Основная часть включает теоретическую часть, аналитическую часть и проектную.

Теоретическая часть представляет обобщенный материал по теме курсовой работы; аналитическая часть содержит маркетинг методов исследования и анализ предметной области; проектная часть содержит основные проектные решения , рекомендации и обоснование выводов по работе.

В заключении обобщаются основные результаты работы, отмечаются перспективы развития проекта.

Важнейшим требованием, предъявляемым к курсовой работе, является самостоятельный характер ее выполнения. Оформление пояснительной записки курсовой работы должно осуществляться в строгом соответствии со стандартом ФГБОУ ВО «БрГУ» «Оформление пояснительной записки учебной работы» СМК СТП 1.4-01-2005.

Пояснительная записка должна быть выполнена аккуратно, без исправлений. Объем Пояснительной записки должен составлять 40–45 страниц.

Законченная работа представляется преподавателю для проверки. В случае выявления при проверке ошибок и неточностей, обучающийся допускается к защите курсовой работы только после их устранения.

Защита курсовой работы предусматривает вопросы по основным разделам работы с целью выявления уровня сформированных компетенций и самостоятельности выполнения.

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

- Microsoft Windows Professional Russian
- Microsoft Office Russian
- Антивирусное программное обеспечение Kaspersky Security
- Справочно-правовая система «Консультант Плюс»
- Bpmn.io
- Delphi Community Edition

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Вид занятия (*Лк, ПЗ, СР…)*	***Наименование аудитории***	***Перечень основного оборудования***	***№ Лк, ПЗ***
1	**3**	**4**	**5**
Лк	Лекционная аудитория (мультимедийный класс)	Интерактивная доска SMART Board 680i2/Unifl, Интерактивный планшет Wacom PL-720, Колонки Microlab Solo-7C, Ноутбук Samsung R610<NP-R610-FS08>, Телевизор плазменный Samsung 63 PS-63A756T1M	Лк № 3,5-7
ПЗ	Дисплейный класс	Системный блок AMD A10-7800 Radeon R7 (12 шт.), Системный блок для слабовидящих пользователей AMD A10-7850K (1 шт.), Монитор Philips233 V5QHABP (13 шт.)	ЛР №№ 1-4
ЛР	Дисплейный класс	Системный блок AMD A10-7800 Radeon R7 (12 шт.), Системный блок для слабовидящих пользователей AMD A10-7850K (1 шт.), Монитор Philips233 V5QHABP (13 шт.)	

КР	Дисплейный класс	Системный блок AMD A10-7800 Radeon R7 (12 шт.), Системный блок для слабовидящих пользователей AMD A10-7850K (1 шт.), Монитор Philips233 V5QHABP (13 шт.)	
	Читальный зал №1	Оборудование 10 ПК i5-2500/H67/4Gb(монитор TFT19 Samsung); принтер HP LaserJet P2055D	
СР	Читальный зал №1	Оборудование 10 ПК i5-2500/H67/4Gb(монитор TFT19 Samsung); принтер HP LaserJet P2055D	-

Приложение 1

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ПК-2	способность разрабатывать, внедрять и адаптировать прикладное программное обеспечение	**1.** Основы разработки программных приложений	**1.1** Информационные технологии. Жизненный цикл информационной системы **1.2** Модели жизненного цикла **1.3** Основные принципы проектирования **1.4** Классификация моделей информационной системы	вопросы к экзамену 1.1-1.5
ПК-7	способность проводить описание прикладных процессов и информационного обеспечения решения прикладных задач	**1.** Основы разработки программных приложений	**1.1** Информационные технологии. Жизненный цикл информационной системы **1.2** Модели жизненного цикла **1.3** Основные принципы проектирования **1.4** Классификация моделей информационной системы	вопросы к экзамену 1.6-1.8
		2 Технологии и подходы к проектированию программных приложений	**2.1** CASE-технологии анализа и проектирования **2.2** Сущность структурного анализа и проектирования **2.3** Разработка функциональной модели	вопросы к экзамену 2.1-2.20
		3. Разработка информационной модели	**3.1** Основы проектирования баз данных **3.2** Концептуальное проектирование с использованием методологии IDEF1X **3.3** Логическое проектирование с использованием методологии IDEF1X **3.4** Физическое проектирование с использованием методологии IDEF1X	вопросы к экзамену 3.1-3.28

ПК-8	способность программировать приложения и создавать программные прототипы решения прикладных задач	**1.** Основы разработки программных приложений	**1.1** Информационные технологии. Жизненный цикл информационной системы **1.2** Модели жизненного цикла **1.3** Основные принципы проектирования **1.4** Классификация моделей информационной системы	вопросы к экзамену 1.9-1.10
		4. Основы объектно-ориентированного подхода к проектированию и разработке программ	**4.1** Объектно-ориентированное проектирование программ **4.2** Основные принципы объектно-ориентированного программирования. Классы и объекты, поля, свойства, методы, события. Конструкторы и деструкторы. **4.3** Проект, файлы, входящие в состав проекта. **4.4** Форма: свойства и методы формы, события, организация, реакция на них **4.5** Визуальные компоненты, использование библиотеки VCL/ **4.6** Событие, обработчик события, создание и использование **4.7** Разработка графического интерфейса. Развитые элементы интерфейса. **4.8** Компоненты для ввода, отображения, редактирования и вывода информации **4.9** Элементы управления на форме. Работа с меню: главное контекстное, системное **4.10** Файлы, окна диалога работы с файлами. Настройка окон диалога**.**	вопросы к экзамену 4.1-4.10

2. Экзаменационные вопросы

№ п/п	Компетенции		ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ	№ и наименование раздела
	Код	Определение		
1	2	3	4	5
1.	ПК-2	способность разрабатывать, внедрять и адаптировать прикладное программное обеспечение	**1.1** Классификация моделей жизненного цикла. **1.2** Инкрементная стратегия. **1.3** Спиральная стратегия. **1.4** Сравнительный анализ моделей. **1.5** Методологии, поддерживающие спиральную модель.	**1.** Основы разработки программных приложений

2.	ПК-7	способность проводить описание прикладных процессов и информационного обеспечения решения прикладных задач	**1.6** Особенности анализа и проектирования крупных систем. **1.7** Документы, содержащие требования на разработку системы. **1.8** Виды обеспечения информационной системы.	**1.** Основы разработки программных приложений
			2.1 CASE-технологии анализа и проектирования ИС. Назначение и основные возможности CASE-средств. **2.2** Модели ИС. Принципы построения моделей. **2.3** Сущность структурного подхода к анализу и проектированию ИС. **2.4** Краткая характеристика методологий структурного анализа и проектирования ИС. Основные принципы проектирования **2.5** Назначение и состав методологии SADT (IDEF0) **2.6** Методология IDEF0. Виды диаграмм. **2.7** Методология IDEF0. Элементы диаграмм. **2.8** Методология IDEF0. ICOM-коды. **2.9** Методология IDEF0. Модели AS-IS, TO-BE и SHOULD-BE. **2.10** Методология IDEF0. Правила и рекомендации построения диаграмм **2.11** Элементы графической нотации IDEF0 **2.12** Типы связей между работами **2.13** ICOM-коды **2.14** Элементы графической нотации DFD **2.15** Правила и рекомендации построения DFD **2.16** Диаграммы потоков данных. Назначение. **2.17** Диаграммы потоков данных. Элементы диаграмм. **2.18** Диаграммы потоков данных. Расширение DFD для систем реального времени. **2.19** ERD. Назначение и основные элементы моделей. **2.20** Возможности современных CASE-средств моделирования данных.	**2.** Технологии и подходы к проектированию программных приложений

			3.1 Основные этапы проектирования баз данных. **3.2** Основные элементы диаграмм «сущность–связь». **3.3** Основные возможности CASE-средств, поддерживающих построение информационных моделей. **3.4** Отличие независимой сущности от зависимой **3.5** Виды атрибутов. **3.6** Домен. Ключ. Типы ключей и их характеристика **3.7** Основные характеристики связи между сущностями. **3.8** Методология IDEF1X. Элементы диаграмм. **3.9** Концептуальное проектирование БД. Стадии. **3.10** Концептуальное проектирование БД. Сущности. **3.11** Концептуальное проектирование БД. Связи. **3.12** Концептуальное проектирование БД. Атрибуты. **3.13** Концептуальное проектирование БД. Ключи. **3.14** Концептуальное проектирование БД. Суперклассы и подклассы. **3.15** Цель логического проектирования БД. Стадии. **3.16** Логическое проектирование БД. Удаление элементов, не отвечающих реляционной модели данных. **3.17** Логическое проектирование БД. Определение требований поддержки целостности данных. **3.18** Основные элементы концептуальной модели, которые могут не отвечать реляционной модели данных. **3.19** Определение понятия «нормализация». **3.20** Первые четыре нормальные формы. **3.21** Полная функциональная и транзитивная зависимости. **3.22** Ограничения целостности. **3.23** Триггер. Действия, вызывающие срабатывание триггера **3.24** Основные стратегии поддержания целостности при помощи триггеров на примере удаления. **3.25** Цель физического проектирования. Стадии. **3.26** Денормализация данных? **3.27** Перечислите основные способы денормализации данных и дайте их характеристику. **3.28** Назовите основные механизмы защиты, применяемые в реляционных базах данных	**3.** Разработка информационной модели

3.	ПК-8	способность программировать приложения и создавать программные прототипы решения прикладных задач	**1.9** Проектированием системы, основные принципы проектирования. **1.10** Классификация моделей системы по отображаемому аспекту баз данных	**1.** Основы разработки программных приложений
			4.1 Варианты представления моделей объектно-ориентированного проектирования программных средств **4.2** Системные подходы к проектированию приложений **4.3** Особенности объектно-ориентированного проектирования программ **4.4** Проблемы объектно-ориентированного проектирования программ и подходы к их решению. **4.5** Основные принципы объектно-ориентированного программирования. Классы и объекты, поля, свойства, методы, события. Конструкторы и деструкторы. **4.6** Проект, файлы, входящие в состав проекта. **4.7** Форма: свойства и методы формы, события, организация, реакция на них **4.8** Визуальные компоненты, использование библиотеки VCL/ **4.9** Событие, обработчик события, создание и использование **4.10** Разработка графического интерфейса. Развитые элементы интерфейса. **4.11** Компоненты для ввода, отображения, редактирования и вывода информации **4.12** Элементы управления на форме. Работа с меню: главное контекстное, системное **4.13** Файлы, окна диалога работы с файлами. Настройка окон диалога.	**4.** Основы объектно-ориентированного подхода к проектированию и разработке программ

3. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
Знать *(ПК-2):* − состав функциональных и обеспечивающих подсистем ИС; − основные подходы и методы проектирования и создания ИС; − основные среды для разработки программного обеспечения; *(ПК-7):* − методы анализа и моделирования бизнес-процессов; − основные стандарты, технологии и нотации моделирования бизнес-процессов; − инструментальные системы, используемые для описания и анализа бизнес-процессов;	**отлично**	Оценка «**отлично**» выставляется в случае, если студент демонстрирует: − всестороннее систематическое знание основных положений теории разработки программных приложений; − правильное выполнение лабораторных заданий, направленных на использование глубоких теоретических знаний и овладение навыками применения современных методов сбора, обработки и анализа данных и разработки программных приложений; − владеет современными методами проектирования и разработки программных приложений.

(ПК-8): – модели и процессы жизненного цикла; – стадии создания ИС; **Уметь** *(ПК-2):* − проводить формализацию и реализацию решения прикладных задач; − моделировать схемы баз данных; − внедрять и адаптировать прикладное программное обеспечение; *(ПК-7):* − моделировать, анализировать и совершенствовать бизнес-процессы с использованием изученных стандартов, технологий и нотаций моделирования; − рецензировать модель бизнес-процесса; *(ПК-8):* − выполнять работы на всех стадиях жизненного цикла ИС; − разрабатывать концептуальную модель предметной области − выполнять работы по созданию технической документации; Владеть *(ПК-2):* − современными языками программирования, методиками разработки и внедрения прикладного программного обеспечения; *(ПК-7):* − практическими навыками моделирования, анализа и документирования бизнес-процессов с помощью инструментальных сред; − терминологией из области моделирования бизнес-процессов; − методами построения, анализа и документирования моделей бизнес-процессов; − навыками работы с инструментальными средствами моделирования предметной области, прикладных и информационных процессов; (ПК-8): − современными языками программирования, методами разработки программных прототипов решения прикладных задач.	**хорошо**	Оценка «**хорошо**» выставляется в случае, если студент демонстрирует: − недостаточное полное знание основных положений теории разработки программных приложений; − выполнение с несущественными ошибками лабораторных заданий, направленных на использование глубоких теоретических знаний и овладение навыками применения современных методов сбора, обработки и анализа данных и разработки программных приложений; − не в полной мере владеет современными методами проектирования и разработки программных приложений.
	удовлетворительно	Оценка «**удовлетворительно**» выставляется в случае, если студент демонстрирует: − частичное знание основных положений теории разработки программных приложений; − частичное выполнение с ошибками лабораторных заданий, направленных на использование глубоких теоретических знаний и овладение навыками применения современных методов сбора, обработки и анализа данных и разработки программных приложений; − не в полной мере владеет современными методами проектирования и разработки программных приложений.
	неудовлетворительно	Оценка «**неудовлетворительно**» выставляется в случае, если студент демонстрирует: − существенные пробелы в знании основных положений теории разработки программных приложений; − принципиальные ошибки при выполнении лабораторных заданий, направленных на использование глубоких теоретических знаний и овладение навыками применения современных методов сбора, обработки и анализа данных и разработки программных приложений; − не владеет современными методами проектирования и разработки программных приложений.

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и опыта деятельности

Цель и задачи дисциплины Б1.В.09 Разработка программных приложений представлены в разделе 1 настоящей рабочей программы. Место дисциплины в структуре образовательной программы представлено в разделе 2 настоящей рабочей программы. Распределение объема дисциплины по формам обучения с указанием видов учебных занятий представлено в разделе 3 настоящей рабочей программы. Содержание дисциплины указано в разделе 4 настоящей рабочей программы.

Учебно-методические материалы для самостоятельной работы студентов по дисциплине находятся в свободном доступе в соответствии с разделом 6 настоящей рабочей программы.

При изучении дисциплины необходимо использовать литературу, указанную в разделе 7 настоящей рабочей программы, а также перечень ресурсов информационно-телекоммуникационной сети «Интернет», представленных в разделе 8 настоящей рабочей программы.

Консультации для студентов по дисциплине проводятся в соответствии с графиком проведения консультаций, представленном на стенде кафедры.

К экзамену допускаются студенты очной, заочной формы обучения, которые теоретически подготовлены по всем вопросам и владеют навыками проектирования и разработки программных приложений.

Информационные технологии, используемые при освоении дисциплины, перечислены в разделе 10 настоящей рабочей программы.

Оценка знаний, умений, навыков осуществляется в процессе промежуточной аттестации обучающихся по дисциплине, которая осуществляется в виде экзамена. Для оценивания знаний, умений, навыков используются ФОС по дисциплине.

Экзамен проводятся в устной форме по выданному преподавателем заданию.

По итогам выполненного задания преподаватель оценивает уровень знаний, умений, навыков. Описание показателей и критериев оценивания компетенций, сформированных по итогам изучения дисциплины, представлено в разделе 3 Приложения 1 настоящей рабочей программы. Основными оценочными средствами при проведении промежуточной аттестации являются вопросы к экзамену.

Приложение 2

АННОТАЦИЯ
рабочей программы дисциплины

Разработка программных приложений

1. Цель и задачи дисциплины

Целью изучения дисциплины является: формирование у обучающихся практических навыков по разработке программного обеспечения (ПО) для решения экономических и расчетных задач с применением современных методов и технологий программирования.

Задачами изучения дисциплины являются:

− знакомство с основными концепциями разработки приложений информационных систем и получение практических навыков в работе с ними;

− получение практических навыков проектирования и разработки программных приложений.

2. Структура дисциплины

2.1 Распределение трудоемкости по отдельным видам учебных занятий, включая самостоятельную работу: 18 ч. лекции, 18ч. лабораторные работы, самостоятельная работа 72 ч.

Общая трудоемкость дисциплины составляет 144 часа, 4 зачетных единицы

2.2 Основные разделы дисциплины:

1. Основы разработки программных приложений
2. Технологии и подходы к проектированию программных приложений
3. Разработка информационной модели
4. Основы объектно-ориентированного подхода к проектированию и разработке программ

3. Планируемые результаты обучения (перечень компетенций)

Процесс изучения дисциплины направлен на формирование следующих компетенций:

ПК-2 - способность разрабатывать, внедрять и адаптировать прикладное программное обеспечение;

ПК-7 - способность проводить описание прикладных процессов и информационного обеспечения решения прикладных задач;

ПК-8 - способность программировать приложения и создавать программные прототипы решения прикладных задач.

4. Вид промежуточной аттестации: экзамен, КР

Приложение 3

Протокол о дополнениях и изменениях в рабочей программе
на 20___-20___ учебный год

1. В рабочую программу по дисциплине вносятся следующие дополнения:

_____

_____

2. В рабочую программу по дисциплине вносятся следующие изменения:

_____

_____

_____

Протокол заседания кафедры №____ от «___» _____ 20 __ __ г.,
(разработчик)

Заведующий кафедрой _____ _____
(подпись) *(Ф.И.О.)*

Программа составлена в соответствии с федеральным государственным образовательным стандартом высшего образования по направлению подготовки 09.03.03 Прикладная информатика от «12» марта 2015 г. № 207

для набора 2014 года: и учебным планом ФГБОУ ВО «БрГУ» для заочной формы обучения от «03» июля 2018 г. № 413

для набора 2015 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «03» июля 2018 г. № 413, заочной формы обучения от «03» июля 2018 г. № 413

для набора 2016 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «05» мая 2016 г. № 342

для набора 2017 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «06» марта 2017 г. №125, заочной формы обучения от «06» марта 2017 г. №125

для набора 2018 года и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «12» марта 2018 г. № 130, заочной формы обучения от «12» марта 2018 г. № 130

Программу составил:

Вахрушева М.Ю., доцент баз. МиИТ, доцент, к.физ.-мат.н. _____

Рабочая программа рассмотрена и утверждена на заседании базовой кафедры МиИТ

от «19» декабря 2018 г., протокол № 8

И.о. заведующего базовой кафедрой МиИТ _____ Е.И. Луковникова

СОГЛАСОВАНО:

И.о. заведующего выпускающей базовой кафедрой МиИТ _____ Е.И. Луковникова

Директор библиотеки _____ Т.Ф. Сотник

Рабочая программа одобрена методической комиссией факультета ФЭиУ

от «28» декабря 2018 г., протокол № 4

Председатель методической комиссии факультета _____ Е.В. Трапезникова

СОГЛАСОВАНО:

Начальник учебно-методического управления _____ Г.П. Нежевец

Регистрационный №_____