Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования «Амурский государственный университет»

Кафедра Информационных и управляющих систем
(наименование кафедры)

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ

СОВРЕМЕННЫЕ ПРОГРАММНО-ТЕХНИЧЕСКИЕ СРЕДСТВА АВТОМАТИЗИРОВАННЫХ СИСТЕМ НАУЧНЫХ ИССЛЕДОВАНИЙ

(наименование дисциплины)

Основной образовательной программы по направлению подготовки:

230100.68 «Информатика и вычислительная техника» по магистерской программе «Компьютерное моделирование»
(код и наименование направления)

Благовещенск 2011

УМКД разработан _____
(степень, звание, фамилия, имя, отчество разработчиков)
_____
_____

Рассмотрен и рекомендован на заседании кафедры

Протокол заседания кафедры от «____» _____ 201___ г. № _____

Зав. кафедрой _____ /_____ /
(подпись) (И.О. Фамилия)

УТВЕРЖДЕН

Протокол заседания УМСС _____
(указывается название специальности (направления подготовки))

от «____» _____ 201___ г. № _____

Председатель УМСС _____ /_____ /
(подпись) (И.О.Фамилия)

1. Рабочая программа учебной дисциплины

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Амурский государственный университет»

УТВЕРЖДАЮ
Проректор по учебной работе
_____ В.В. Проказин
«___» _____ 20__ г.

РАБОЧАЯ ПРОГРАММА

СОВРЕМЕННЫЕ ПРОГРАММНО-ТЕХНИЧЕСКИЕ СРЕДСТВА АВТОМАТИЗИРОВАННЫХ СИСТЕМ НАУЧНЫХ ИССЛЕДОВАНИЙ

Направление подготовки 230100.68 «Информатика и вычислительная техника»
по магистерской программе «Компьютерное моделирование»
Квалификация (степень) выпускника – магистр
Специальное звание – магистр-инженер

Курс – 1
Семестр – 1
Лекции – 18 час.
Экзамен – нет
Практические (семинарские) занятия – нет
Зачет – 1
Лабораторные занятия – нет
Самостоятельная работа – 90 (час.)
Общая трудоемкость дисциплины – 108 (час.), 3 (з.е.)
Курсовая работа (проект) – нет
Составитель – А.В. Бушманов, доцент, канд. техн. наук
Факультет математики и информатики
Кафедра информационных и управляющих систем

2011 г.

Рабочая программа составлена на основании Федерального государственного образовательного стандарта высшего профессионального образования по направлению подготовки 230100 «Информатика и вычислительная техника» (квалификация (степень) «магистр»)

Рабочая программа обсуждена на заседании кафедры информационных и управляющих систем
«___» _____ 20___ г., протокол № _____
Заведующий кафедрой _____ А.В. Бушманов

Рабочая программа одобрена на заседании учебно-методического совета направления подготовки 230100.68 «Информатика и вычислительная техника»
«___» _____ 20___ г., протокол № _____
Председатель _____ В.В. Еремина

Рабочая программа переутверждена на заседании кафедры информационных и управляющих систем
«___» _____ 20___ г., протокол № _____
Заведующий кафедрой _____ А.В. Бушманов

СОГЛАСОВАНО
Начальник учебно-методического управления
_____
«___» _____ 20___ г.

СОГЛАСОВАНО
Председатель учебно-методического совета факультета
_____ С.Г. Самохвалова
«___» _____ 20___ г.

СОГЛАСОВАНО
Заведующий выпускающей кафедрой
_____ А.В. Бушманов
«___» _____ 20___ г.

СОГЛАСОВАНО
Директор научной библиотеки
_____ Л.А. Проказина
«___» _____ 20___ г.

1 ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Цель дисциплины: обзор и использование современных программно-технических средств автоматизированных систем для научных исследований.

Задачи дисциплины:

– изучение современных программно-технических средств автоматизированных систем для научных исследований;

– формирование устойчивых навыков практического использования современных программно-технических средств автоматизированных систем для научных исследований.

2 МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ООП ВПО

Дисциплина относится к вариативным дисциплинам цикла профессиональных дисциплин (М2.В2) Федерального государственного образовательного стандарта высшего профессионального образования по направлению подготовки 230100 «Информатика и вычислительная техника» (квалификация (степень) «магистр»).

Для успешного освоения данной дисциплины необходимы знания, умения и навыки, приобретенные в результате освоения дисциплин базовой части математического и естественнонаучного цикла (Б.2) Федерального государственного образовательного стандарта высшего профессионального образования по направлению подготовки 230100 «Информатика и вычислительная техника» (квалификация (степень) «бакалавр»): информатика; информационные технологии; математическая статистика и случайные процессы; вычислительная математика.

Знания, умения и навыки, приобретенные в результате освоения данной дисциплины необходимы для использования типовых программных продуктов, ориентированных на решение научных, проектных и технологических задач, а также для выполнения научно-исследовательской работы написании магистерской диссертации (М.3).

3 КОМПЕТЕНЦИИ ОБУЧАЮЩЕГОСЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования (ПК-3, ПК-6):

1) Знать: современные программно-технические средства автоматизированных систем для научных исследований.

2) Уметь: использовать современные программно-технические средства автоматизированных систем в научных исследованиях.

3) Владеть: методами научного мышления.

4 СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Общая трудоемкость дисциплины составляет 3.0 зачетные единицы, 108 часов.

№ п/п	Раздел дисциплины	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость в часах				Формы текущего контроля успеваемости Форма промежуточной аттестации
				Лек	Пр	Лаб	Сам	
1	Структура, содержание и задачи курса, его связь с другими дисциплинами.	9	1-2	2	0	0	2	Собеседование
2	Программное обеспечение автоматизированных систем	9	3-4	2	0	0	6	Собеседование
			5-6	2	0	0	6	Собеседование

3	CASE-системы в исследованиях и проектировании	9	7-8	2	0	0	8	Собеседование
			9-10	2	0	0	10	Собеседование
4	Методология объектно-ориентированного проектирования	9	11-12	2	0	0	8	Собеседование
			13-14	2	0	0	14	Собеседование
5	Разработка моделей с помощью языка UML	9	15-16	2	0	0	16	Собеседование
			17-18	2	0	0	20	Собеседование
6	Всего по разделам	9	1-18	18	0	0	90	Зачет

5 СОДЕРЖАНИЕ РАЗДЕЛОВ И ТЕМ ДИСЦИПЛИНЫ

5.1 Лекции

5.1.1 Лекция 1. Критерии оценки и выбора CASE-средств.

5.1.2 Лекция 2. Методология IDEF0. Модели AS-IS и TO-BE.

5.1.3 Лекция 3. Разработка модели с использованием нотации IDEFX1.

5.1.4 Лекция 4. Модель информационного пространства с помощью CASE-средств верхнего уровня BPWIN.

5.1.5 Лекция 5. Модель информационного пространства с помощью CASE-средств верхнего уровня ERWIN.

5.1.6 Лекция 6. ООП. Декомпозиция. Отношение между классами. Иерархия классов. Объекты и классы. Абстрагирование и обобщение.

5.1.7 Лекция 7. ООП. Ограниченность доступа. Модульность.

5.1.8 Лекция 8. Пакеты и классы. Уточнение методов и свойств классов.

5.1.9 Лекция 9. Кодогенерация проекта в Delphi.

6 САМОСТОЯТЕЛЬНАЯ РАБОТА

№ п/п	Раздел дисциплины	Форма (вид) самостоятельной работы	Трудоемкость в часах
1	Критерии оценки и выбора CASE-средств.	Реферат	16
2	Программное обеспечение автоматизированных систем.	Реферат	16
3	CASE-системы в исследованиях и проектировании.	Реферат	18
4	Методология объектно-ориентированного проектирования.	Реферат	18
5	Разработка моделей с помощью языка UML.	Реферат	22

7 МАТРИЦА КОМПЕТЕНЦИЙ

№ п/п	Раздел дисциплины	Компетенции		Общее кол-во компетенций
		ПК3	ПК6	
1	Критерии оценки и выбора CASE-средств.	+		1
2	Программное обеспечение автоматизированных систем.	+		1
3	CASE-системы в исследованиях и проектировании.		+	1

4	Методология объектно-ориентированного проектирования.		+	1
5	Разработка моделей с помощью языка UML.		+	1

8 ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

Образовательный процесс по дисциплине строится на основе комбинации следующих образовательных технологий.

Интегральную модель образовательного процесса по дисциплине формируют технологии методологического уровня: модульно-рейтинговое обучение, технология поэтапного формирования умственных действий, технология развивающего обучения, элементы технологии развития критического мышления.

Реализация данной модели предполагает использование следующих технологий стратегического уровня (задающих организационные формы взаимодействия субъектов образовательного процесса), осуществляемых с использованием определенных тактических процедур:

– лабораторные (углубление знаний, полученных ранее на теоретических занятиях, решение задач);

– тренинговые (формирование определенных умений и навыков, формирование алгоритмического мышления);

– активизации познавательной деятельности (приемы технологии развития критического мышления через чтение и письмо, работа с литературой, подготовка презентаций по темам домашних работ);

– самоуправления (самостоятельная работа студентов, самостоятельное изучение материала).

Рекомендуется использование информационных технологий при организации коммуникации со студентами для представления информации, выдачи рекомендаций и консультирования по оперативным вопросам (электронная почта), использование мультимедиа-средств при проведении лекционных и практических занятий.

9 ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ, ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ И УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

9.1 Оценочные средства для текущего контроля успеваемости

9.1.1 Контрольные вопросы допуска к выполнению лабораторных работ

9.1.2 Отчеты о выполнении индивидуальных вариантов заданий лабораторных работ

9.2 Оценочные средства для промежуточной аттестации

Вопросы к зачету:

9.2.1 Обзор программных и технических средств автоматизированных систем.

9.2.2 Прикладное и системное обеспечение автоматизированных систем.

9.2.3 Существующие методы разработки, средств и технологий применения программного обеспечения ВТ и АС в научных исследованиях. Автоматизированные системы в проектно-конструкторской деятельности, управлении технологическими, экономическими, социальными системами и в гуманитарных областях деятельности человека.

9.2.4 Классификация CASE-систем по функциональному назначению.

9.2.5 CASE-технология как совокупность методологий анализа, проектирования, разработки и сопровождения сложных систем.

9.2.6 Поддержка CASE-технологии комплексом взаимоувязанных средств автоматизации.

9.2.7 Методология объектно-ориентированного анализа и проектирования программного обеспечения как разновидность реализации системного подхода.
9.2.8 Методологии объектно-ориентированного анализа и проектирования (ООАП).
9.2.9 Объектно-ориентированные системы.
9.2.10 Подходы к проектированию программ в целом. Родственные методологии.
9.2.11 Пакеты, как основные группирующие сущности, с помощью которых можно организовать модель UML.
9.2.12 Спецификация UML, правила для корректного построения моделей.
9.2.13 Избыточность языка UML.
9.2.14 Подходы к трансформации UML-моделей.
9.2.15 Разработка приложений реального времени с помощью UML.
9.2.16 Аспекты использования языка UML.

9.3 Учебно-методическое обеспечение самостоятельной работы
9.3.1 Карточки с заданиями и методическими указаниями по выполнению лабораторных работ
9.3.2 СТО СМК 4.2.3.05-2011. Стандарт ФГБОУОВПО «АмГУ». Оформление выпускных квалификационных и курсовых работ (проектов), 2011. – 95 с.

10 УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

а) основная литература:

10.1 Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.
10.2 Советов Б.Я. Информационные технологии: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.
10.3 Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. - М.: ФОРУМ: ИНФРА-М, 2007.-416 с. (10 экз.)

б) дополнительная литература:

10.4 Бобровский С.И. Технологии Delphi 2006. Новые возможности/ С. И. Бобровский. - СПб.: Питер, 2006.-288 с.
10.5 Информационно-измерительная техника и технологии: Учеб. для вузов: Рек. Мин.обр. РФ/ В. И. Калашников, С.В.Нефедов, А.Б.Путилин; ред. Г.Г.Раннев. -М.: Высш. шк., 2002.-456 с.
10.6 Семенов А.С. Информационные технологии: объектно-ориентированное моделирование: Анализ и проектирование производственных систем: учеб.пособие: рек. УМО по обр./А.С.Семенов. -М.: Изд-во Моск.гос.технол. ун-та Станкин, 2001.-65 с..
10.7 Холмогоров В. Основы веб-мастерства: Учеб. курс/ В. Холмогоров. -2-е изд.. - СПб.: Питер, 2003.-317 с.
10.8 Хорошилов А.В. Мировые информационные ресурсы: Учеб.пособие: Рек. УМО по обр./ А.В. Хорошилов , С.Н.Селетков. -СПб.: Питер, 2004.-176 с.
10.9 Городняя Л.В. Основы функционального программирования : Курс лекций: учеб. пособие: Рек. УМО вузов/ Л.В.Городняя. -М.: Интернет- Ун-т Информ. Технологий, 2004.-273 с.
10.10 Кватрани Т. Визуальное моделирование с помощью Rational Rose 2002 и UML: Пер. с англ./ Т. Кватрани. -М.: Вильямс, 2003.-187 с.

в) периодические издания:

10.10 Программирование
10.11 Программные продукты и системы
10.12 Информатика и системы управления
10.13 Computer – IEEE Computer Magazine

г) программное обеспечение и Интернет-ресурсы:
Свободно распространяемое программное обеспечение

№ п/п	Наименование ресурса	Характеристика
1	рее http://www.iqlib.ru	Интернет библиотека образовательных изданий, в которой собраны электронные учебники, справочные и учебные пособия. Удобный поиск по ключевым словам, отдельным темам и отраслям знаний.
2	http://www.intuit.ru	Интернет-университет информационных технологий, в котором вобраны электронные и видео-курсы по отраслям знаний
3	http://amursu.ru	Сайт АмГУ, Библиотека – электронная библиотека АмГУ
4	http://www.biblioclub.ru	Электронная библиотечная система «Университетская библиотека – online»: специализируется на учебных материалах для ВУЗов по научно-гуманитарной тематике, а так же содержит материалы по точным и естественным наукам

11 МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

11.1 Лекционная аудитория, оборудованная мультимедийными средствами

12 РЕЙТИНГОВАЯ ОЦЕНКА ЗНАНИЙ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ

Семестровый модуль дисциплины						
№ п/п	Раздел дисциплины	Виды контроля	Сроки выполнения (недели)	Максимальное кол-во баллов	Посещение, активность на занятиях	Максимальное кол-во баллов за модуль
1	Критерии оценки и выбора CASE-средств.	ЛР № 1	1-2	5	1	6
2	Программное обеспечение автоматизированных систем.	ЛР № 2	3-4	5	1	12
		ЛР № 3	5-6	5	1	
3	CASE-системы в исследованиях и проектировании.	ЛР № 4	7-8	5	1	12
		ЛР № 5	9-10	5	1	
4	Методология объектно-ориентированного проектирования.	ЛР № 6	11-12	5	1	12
		ЛР № 7	13-14	5	1	
5	Разработка моделей с помощью языка UML.	ЛР № 8	15-16	5	1	18
		ЛР № 9	17-18	5	1	
6	Промежуточная аттестация	зачет	18	6	0	40
Итого						100

2 Краткое изложение программного материала

Лекция 1. Критерии оценки и выбора CASE-средств.

1. Общие сведения

Модель процесса оценки и выбора, рассматриваемая ниже (рисунок 1), описывает наиболее общую ситуацию оценки и выбора, а также показывает зависимость между ними. Как можно видеть, оценка и выбор могут выполняться независимо друг от друга или вместе, каждый из этих процессов требует применения определенных критериев.

Процесс оценки и выбора может преследовать несколько целей, включая одну или более из следующих:

- оценка нескольких CASE-средств и выбор одного или более из них;
- оценка одного или более CASE-средств и сохранение результатов для последующего использования;
- выбор одного или более CASE-средств с использованием результатов предыдущих оценок.

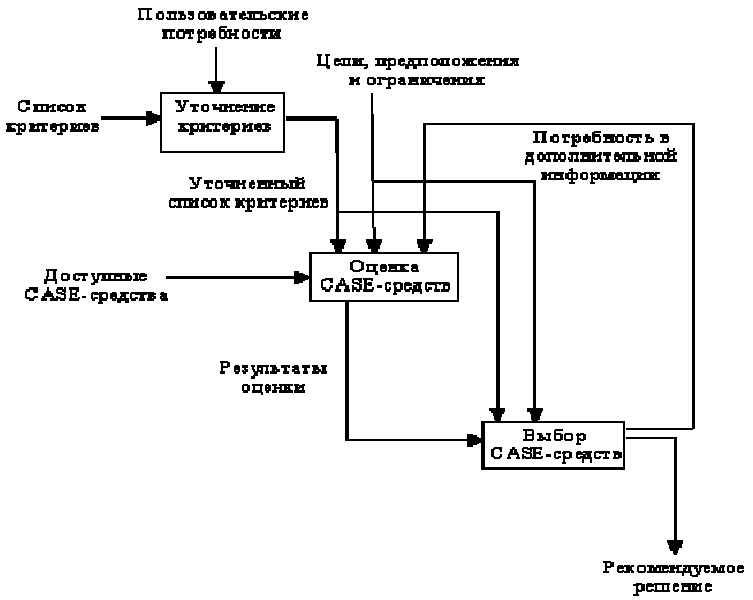


Рис. 1. Модель процесса оценки и выбора

Как видно из рисунка, входной информацией для процесса оценки является:

- определение пользовательских потребностей;
- цели и ограничения проекта;
- данные о доступных CASE-средствах;
- список критериев, используемых в процессе оценки.

Результаты оценки могут включать результаты предыдущих оценок. При этом не следует забывать, что набор критериев, использовавшихся при предыдущей оценке, должен быть совместимым с текущим набором. Конкретный вариант реализации процесса (оценка и выбор, оценка для будущего выбора или выбор, основанный на предыдущих оценках) определяется перечисленными выше целями.

Элементы процесса включают:

- цели, предположения и ограничения, которые могут уточняться в ходе процесса;
- потребности пользователей, отражающие количественные и качественные требования пользователей к CASE-средствам;
- критерии, определяющие набор параметров, в соответствии с которыми производится оценка и принятие решения о выборе;
- формализованные результаты оценок одного или более средств;
- рекомендуемое решение (обычно либо решение о выборе, либо дальнейшая оценка).

Процесс оценки и/или выбора может быть начат только тогда, когда лицо, группа или организация полностью определила для себя конкретные потребности и формализовала их в виде количественных и качественных требований в заданной предметной области. Термин "пользовательские требования" далее означает именно такие формализованные требования.

Пользователь должен определить конкретный порядок действий и принятия решений с любыми необходимыми итерациями. Например, процесс может быть представлен в виде дерева решений с его последовательным обходом и выбором подмножеств кандидатов для более детальной оценки. Описание последовательности действий должно определять поток данных между ними.

Определение списка критериев основано на пользовательских требованиях и включает:

- выбор критериев для использования из приведенного далее перечня;
- определение дополнительных критериев;
- определение области использования каждого критерия (оценка, выбор или оба процесса);
- определение одной или более метрик для каждого критерия оценки;
- назначение веса каждому критерию при выборе.

Процесс оценки

Целью процесса оценки является определение функциональности и качества CASE-средств для последующего выбора. Оценка выполняется в соответствии с конкретными критериями, ее результаты включают как объективные, так и субъективные данные по каждому средству.

Процесс оценки включает следующие действия:

- формулировка задачи оценки, включая информацию о цели и масштабах оценки;
- определение критериев оценки, вытекающее из определения задачи;
- определение средств-кандидатов путем просмотра списка кандидатов и анализа информации о конкретных средствах;
- оценка средств-кандидатов в контексте выбранных критериев. Необходимые для этого данные могут быть получены путем анализа самих средств и их документации, опроса пользователей, работы с демо-версиями, выполнения тестовых примеров, экспериментального применения средств и анализа результатов предшествующих оценок;
- подготовка отчета по результатам оценки.

Одним из важнейших критериев в процессе оценки может быть потенциальная возможность интеграции каждого из средств-кандидатов с другими средствами, уже находящимися в эксплуатации или планируемыми к использованию в данной организации.

Масштаб оценки должен устанавливать требуемый уровень детализации, необходимые ресурсы и степень применимости ее результатов. Например, оценка должна выполняться для набора из одного или более конкретных CASE-средств; CASE-средств, поддерживающих один или более конкретных процессов создания и сопровождения ПО или CASE-средств, поддерживающих один или более проектов или типов проектов.

Список CASE-средств - возможных кандидатов формируется из различных источников: обзоров рынка ПО, информации поставщиков, обзоров CASE-средств и других подобных публикаций.

Следующим шагом является получение информации о CASE-средствах или получение их самих или и то, и другое. Эта информация может состоять из оценок независимых экспертов, сообщений и отчетов поставщиков CASE-средств, результатов демонстрации возможностей CASE-средств со стороны поставщиков и информации, полученной непосредственно от реальных пользователей. Сами CASE-средства могут быть получены путем приобретения, в виде оценочной копии или другими способами.

Оценка и накопление соответствующих данных может выполняться следующими способами:

- анализ CASE-средств и документации поставщика;
- опрос реальных пользователей;
- анализ результатов проектов, использовавших данные CASE-средства;
- просмотр демонстраций и опрос демонстраторов;
- выполнение тестовых примеров;
- применение CASE-средств в пилотных проектах;

- анализ любых доступных результатов предыдущих оценок.

Существуют как объективные, так и субъективные критерии. Результаты оценки в соответствии с конкретным критерием могут быть двоичными, находиться в некотором числовом диапазоне, представлять собой просто числовое значение или иметь какую-либо другую форму.

Для объективных критериев оценка должна выполняться путем воспроизводимой процедуры, чтобы любой другой специалист, выполняющий оценку, мог получить такие же результаты. Если используются тестовые примеры, их набор должен быть заранее определен, унифицирован и документирован.

По субъективным критериям CASE-средство должно оцениваться группой специалистов, использующих одни и те же критерии. Необходимый уровень опыта специалистов или групп должен быть заранее определен.

Результаты оценки должны быть стандартным образом документированы (для облегчения последующего использования) и, при необходимости, утверждены.

Отчет по результатам оценки должен содержать следующую информацию:

- *введение*. Общий обзор процесса и перечень основных результатов;
- *предпосылки*. Цель оценки и желаемые результаты, период времени, в течение которого выполнялась оценка, определение ролей и соответствующего опыта специалистов, выполнявших оценку;
- *подход к оценке*. Описание общего подхода, включая полученные CASE-средства, информацию, определяющую контекст и масштаб оценки, а также любые предположения и ограничения;
- *информация о CASE-средствах*. Она должна включать следующее: 1) наименование CASE-средства; 2) версию CASE-средства; 3) данные о поставщике, включая контактный адрес и телефон; 4) конфигурацию технических средств; 5) стоимостные данные; 6) описание CASE-средства, включающее поддерживаемые данным средством процессы создания и сопровождения ПО, программную среду CASE-средства (в частности, поддерживаемые языки программирования, операционные системы, совместимость с базами данных), функции CASE-средства, входные/выходные данные и область применения;
- *этапы оценки*. Конкретные действия, выполняемые в процессе оценки, должны быть описаны со степенью детализации, необходимой как для понимания масштаба и глубины оценки, так и для ее повторения при необходимости;
- *конкретные результаты*. Результаты оценки должны быть представлены в терминах критериев оценки. В тех случаях, когда отчет охватывает целый ряд CASE-средств или результаты данной оценки будут сопоставляться с аналогичными результатами других оценок, необходимо обратить особое внимание на формат представления результатов, способствующий такому сравнению. Субъективные результаты должны быть отделены от объективных и должны сопровождаться необходимыми пояснениями;
- *выводы и заключения;*
- *приложения.* Формулировка задачи оценки и уточненный список критериев.

Процесс выбора

Процессы оценки и выбора тесно взаимосвязаны друг с другом. По результатам оценки цели выбора и/или критерии выбора и их веса могут потребовать модификации. В таких случаях может потребоваться повторная оценка. Когда анализируются окончательные результаты оценки и к ним применяются критерии выбора, может быть рекомендовано приобретение CASE-средства или набора CASE-средств. Альтернативой может быть отсутствие адекватных CASE-средств, в этом случае рекомендуется разработать новое CASE-средство, модифицировать существующее или отказаться от внедрения.

Процесс выбора тесно взаимосвязан с процессом оценки и включает следующие действия:

- формулировка задач выбора, включая цели, предположения и ограничения;

- выполнение всех необходимых действий по выбору, включая определение и ранжирование критериев, определение средств-кандидатов, сбор необходимых данных и применение ранжированных критериев к результатам оценки для определения средств с наилучшими показателями. Для многих пользователей важным критерием выбора является интегрируемость CASE-средства с существующей средой;
- выполнение необходимого количества итераций с тем, чтобы выбрать (или отвергнуть) средства, имеющие сходные показатели;
- подготовка отчета по результатам выбора.

В процессе выбора возможно получение двух результатов:

- рекомендаций по выбору конкретного CASE-средства;
- запроса на получение дополнительной информации к процессу оценки.

Масштаб выбора должен устанавливать требуемый уровень детализации, необходимые ресурсы, график и ожидаемые результаты. Существует ряд параметров, которые могут быть использованы для определения масштаба, включая:

- использование предварительного отбора (например, отбор только средств, работающих на конкретной платформе);
- использование ранее полученных результатов оценки, результатов оценки из внешних источников или комбинации того и другого;

В том случае, если предыдущие оценки выполнялись с использованием различных наборов критериев или выполнялись с использованием конкретных критериев, но различными способами, результаты оценок должны быть представлены в согласованной форме. После завершения данного шага оценка каждого CASE-средства должна быть представлена в рамках единого набора критериев и должна быть непосредственно сопоставима с другими оценками.

Алгоритмы, обычно используемые для выбора, могут быть основаны на масштабе или ранге. Алгоритмы, основанные на масштабе, вычисляют единственное значение для каждого CASE-средства путем умножения веса каждого критерия на его значение (с учетом масштаба) и сложения всех произведений. CASE-средство с наивысшим результатом получает первый ранг. Алгоритмы, основанные на ранге, используют ранжирование CASE-средств - кандидатов по отдельным критериям или группам критериев в соответствии со значениями критериев в заданном масштабе. Затем, аналогично предыдущему, ранги сводятся вместе и вычисляются общие значения рангов.

При анализе результатов выбора предполагается, что процесс выбора завершен, CASE-средство выбрано и рекомендовано к использованию. Тем не менее, может потребоваться более точный анализ для определения степени зависимости значений ключевых критериев от различий в значениях характеристик CASE-средств - кандидатов. Такой анализ позволит определить, насколько результат ранжирования CASE-средств зависит от оптимальности выбора весовых коэффициентов критериев. Он также может использоваться для определения существенных различий между CASE-средствами с очень близкими значениями критериев или рангами.

Если ни одно из CASE-средств не удовлетворяет минимальным критериям, выбор (возможно, вместе с оценкой) может быть повторен для других CASE-средств - кандидатов.

Если различия между самыми предпочтительными кандидатами несущественны, дополнительная информация может быть получена путем повторного выбора (возможно, вместе с оценкой) с использованием дополнительных или других критериев.

Рекомендации по выбору должны быть строго обоснованы. В случае отсутствия адекватных CASE-средств, как было отмечено выше, рекомендуется разработать новое CASE-средство, модифицировать существующее или отказаться от внедрения.

Критерии оценки и выбора

Критерии формируют базис для процессов оценки и выбора и могут принимать различные формы, включая:

- числовые меры в широком диапазоне значений, например, объем требуемой памяти;

- числовые меры в ограниченном диапазоне значений, например, простота освоения, выраженная в баллах от 1 до 5;
- двоичные меры (истина/ложь, да/нет), например, способность генерации документации в формате Postscript;
- меры, которые могут принимать одно или более из конечных множеств значений, например, платформы, для которых поддерживается CASE-средство.

Типичный процесс оценки и/или выбора может использовать набор критериев различных типов.

Структура набора критериев приведена на рисунке 2. Каждый критерий должен быть выбран и адаптирован экспертом с учетом особенностей конкретного процесса. В большинстве случаев только некоторые из множества описанных ниже критериев оказываются приемлемыми для использования, при этом также добавляются дополнительные критерии. Выбор и уточнение набора используемых критериев является критическим шагом в процессе оценки и/или выбора.

Функциональные характеристики

Критерии первого класса предназначены для определения функциональных характеристик CASE-средства. Они в свою очередь подразделяются на ряд групп и подгрупп.

1. Среда функционирования:
   a. Проектная среда:
      - *поддержка процессов жизненного цикла*. Определяет набор процессов ЖЦ, которые поддерживает CASE-средство. Примерами таких процессов являются анализ требований, проектирование, реализация, тестирование и оценка, сопровождение, обеспечение качества, управление конфигурацией и управление проектом, причем они зависят от принятой пользователем модели ЖЦ.
      - *область применения.* Примерами являются системы обработки транзакций, системы реального времени, информационные системы и т.д.
      - *размер поддерживаемых приложений*. Определяет ограничения на такие величины, как количество строк кода, уровней вложенности, размер базы данных, количество элементов данных, количество объектов конфигурационного управления.
   b. ПО/технические средства:
      - *требуемые технические средства*. Оборудование, необходимое для функционирования CASE-средства, включая тип процессора, объем оперативной и дисковой памяти.
      - *поддерживаемые технические средства*. Элементы оборудования, которые могут использоваться CASE-средством, например, устройства ввода/вывода.
      - *требуемое ПО*. ПО, необходимое для функционирования CASE-средства, включая операционные системы и графические оболочки.
      - *поддерживаемое ПО*. Программные продукты, которые могут использоваться CASE-средством.
   c. Технологическая среда:
      - *соответствие стандартам технологической среды*. Такие стандарты касаются языка, базы данных, репозитория, коммуникаций, графического интерфейса пользователя, документации, разработки, управления конфигурацией, безопасности, стандартов обмена информацией и интеграции по данным, по управлению и по пользовательскому интерфейсу.
      - *совместимость с другими средствами*. Способность к взаимодействию с другими средствами, включая непосредственный обмен данными (примерами таких средств являются текстовые процессоры, базы данных и другие CASE-средства). Возможность преобразования репозитория или его части в стандартный формат для обработки другими средствами.

- *поддерживаемая методология*. Набор методов и методик, поддерживаемых CASE-средством. Примерами являются структурный или объектно-ориентированный анализ и проектирование.
- *поддерживаемые языки*. Все языки, используемые CASE-средством. Примерами таких языков являются языки программирования (Кобол, Ада, С), языки баз данных и языки запросов (DDL, SQL), графические языки (Postscript, HPGL), языки спецификации проектных требований и интерфейсы операционных систем (языки управления заданиями).

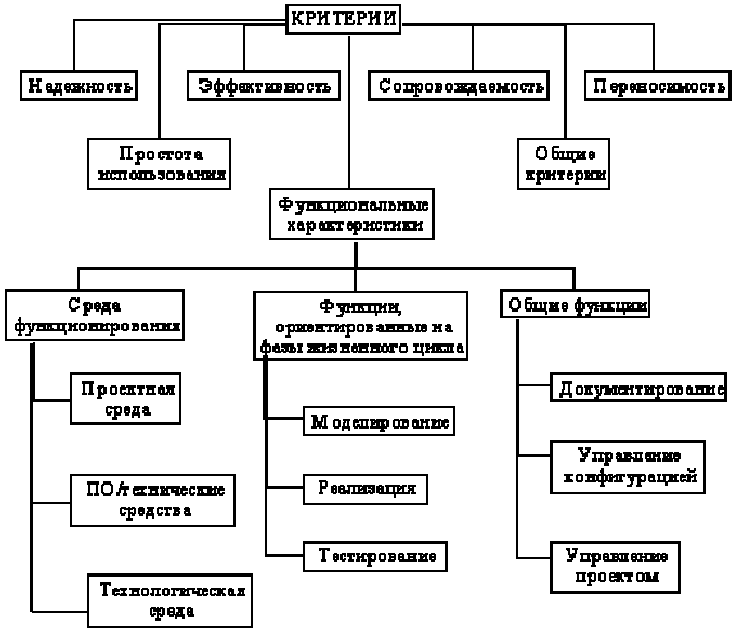


Рис. 2. Структура набора критериев

1. Функции, ориентированные на фазы жизненного цикла:
   a. Моделирование:

Данные критерии определяют способность выполнения функций, необходимых для спецификации требований к ПО и преобразованию их в проект:

- *построение диаграмм*. Возможность создания и редактирования диаграмм различных типов, представляющих интерес для пользователя. Наиболее распространенные типы диаграмм описаны в разделе 2.
- *графический анализ*. Возможность анализа графических объектов, а также хранения и представления проектной информации в графическом представлении. В большинстве случаев графические анализаторы интегрированы со средствами построения диаграмм.
- *ввод и редактирование спецификаций требований и проектных спецификаций*. К спецификациям такого рода относятся описания функций, данных, интерфейсов, структуры, качества, производительности, технических средств, среды, затрат и графиков.
- *язык спецификации требований и проектных спецификаций*. Возможность импорта, экспорта и редактирования спецификаций с использованием формального языка.
- *моделирование данных*. Возможность ввода и редактирования информации, описывающей элементы данных системы и их отношения.
- *моделирование процессов*. Возможность ввода и редактирования информации, описывающей процессы системы и их отношения.
- *проектирование архитектуры ПО*. Проектирование логической структуры ПО - структуры модулей, интерфейсов и др.

- *имитационное моделирование*. Возможность динамического моделирования различных аспектов функционирования системы на основе спецификаций требований и/или проектных спецификаций, включая внешний интерфейс и производительность (например, время отклика, коэффициент использования ресурсов и пропускную способность).
- *прототипирование.* Возможность проектирования и генерации предварительного варианта всей системы или ее отдельных компонент на основе спецификаций требований и/или проектных спецификаций. Прототипирование в основном касается внешнего пользовательского интерфейса и осуществляется при непосредственном участии пользователей.
- *генерация экранных форм*. Возможность генерации экранных форм на основе спецификаций требований и/или проектных спецификаций.
- *возможность трассировки*. Возможность сквозного анализа функционирования системы от спецификации требований до конечных результатов (установления и отслеживания соответствий и связей между функциональными и другими внешними требованиями к ИС, техническими решениями и результатами проектирования). Прямая трассировка (проверка учета всех требований) и обратная трассировка (поиск проектных решений, не связанных ни с какими внешними требованиями).
- *синтаксический и семантический контроль проектных спецификаций.*Контроль синтаксиса диаграмм и типов их элементов, контроль декомпозиции функций, проверка спецификаций на полноту и непротиворечивость.
- *другие виды анализа*. Конкретные дополнительные виды анализа могут включать алгоритмы, потоки данных, нормализацию данных, использование данных, пользовательский интерфейс.
- *автоматизированное проектирование отчетов.*

b. Реализация:

Реализация затрагивает функции, связанные с созданием исполняемых элементов системы (программных кодов) или модификацией существующей системы. Многие из перечисленных ниже критериев зависят от конкретных языков и включают следующие:

- *синтаксически управляемое редактирование*. Возможность ввода и редактирования исходных кодов на одном или нескольких языках с одновременным синтаксическим контролем.
- *генерация кода*. Возможность генерации кодов на одном или нескольких языках на основе проектных спецификаций. Типы генерируемого кода могут включать обычный программный код, схему базы данных, запросы, экраны/меню.
- *компиляция кода.*
- *конвертирование исходного кода*. Возможность преобразования кода из одного языка в другой.
- *анализ надежности*. Возможность количественно оценивать параметры надежности ПО, такие, как количество ошибок и др.
- *реверсный инжиниринг*. Возможность анализа существующих исходных кодов и формирования на их основе проектных спецификаций.
- *реструктуризация исходного кода*. Возможность модификации формата и/или структуры существующего исходного кода.
- *анализ исходного кода*. Примерами такого анализа могут быть определение размера кода, вычисление показателей сложности, генерация перекрестных ссылок и проверка на соответствие стандартам.
- *отладка*. Типичные функции отладки - трассировка программ, выделение узких мест и наиболее часто используемых фрагментов кода и т.д.

c. Тестирование:

Критерии тестирования включают следующие:

- *описание тестов*. Типичные возможности включают генерацию тестовых данных, алгоритмов тестирования, требуемых результатов и т.д.
- *фиксация и повторение действий оператора*. Возможность фиксировать данные, вводимые оператором с помощью клавиатуры, мыши и т.д., редактировать их и воспроизводить в тестовых примерах.
- *автоматический запуск тестовых примеров.*
- *регрессионное тестирование*. Возможность повторения и модификации ранее выполненных тестов для определения различий в системе и/или среде.
- *автоматизированный анализ результатов тестирования*. Типичные возможности включают сравнение ожидаемых и реальных результатов, сравнение файлов, статистический анализ результатов и др.
- *анализ тестового покрытия*. Оснащенность средствами контроля исходного кода и анализ тестового покрытия. Проверяются, в частности, обращения к операторам, процедурам и переменным.
- *анализ производительности*. Возможность анализа производительности программ. Анализируемые параметры производительности могут включать использование центрального процессора, памяти, обращения к определенным элементам данных и/или сегментам кода, временные характеристики и т.д.
- *анализ исключительных ситуаций в процессе тестирования.*
- *динамическое моделирование среды.* В частности, возможность автоматически генерировать моделируемые входные данные системы.

2. Общие функции:

Приведенные ниже критерии определяют функции CASE-средств, охватывающие всю совокупность фаз ЖЦ. Поддержка всех этих функций осуществляется посредством репозитория.

a. Документирование:

- *редактирование текстов и графики*. Возможность вводить и редактировать данные в текстовом и графическом формате.
- *редактирование с помощью форм*. Возможность поддерживать формы, определенные пользователями, вводить и редактировать данные в соответствии с формами.
- *возможности издательских систем.*
- *поддержка функций и форматов гипертекста.*
- *соответствие стандартам документирования.*
- *автоматическое извлечение данных из репозитория и генерация документации по спецификациям пользователя.*

b. Управление конфигурацией:

- *контроль доступа и изменений*. Возможность контроля доступа на физическом уровне к элементам данных и контроля изменений. Контроль доступа включает возможности определения прав доступа к компонентам, а также извлечения элементов данных для модификации, блокировки доступа к ним на время модификации и помещения обратно в репозиторий.
- *отслеживание модификаций*. Фиксация и ведение журнала всех модификаций, внесенных в систему в процессе разработки или сопровождения.
- *управление версиями*. Ведение и контроль данных о версиях системы и всех ее коллективно используемых компонентах.
- *учет состояния объектов конфигурационного управления*. Возможность получения отчетов о всех последовательных версиях, содержимом и состоянии различных объектов конфигурационного управления.

- *генерация версий и модификаций*. Поддержка пользовательского описания последовательности действий, требуемых для формирования версий и модификаций, и автоматическое выполнение этих действий.
- *архивирование*. Возможность автоматического архивирования элементов данных для последующего использования.

с. Управление проектом:

- *управление работами и ресурсами*. Контроль и управление процессом проектирования ИС в терминах структуры заданий и назначения исполнителей, последовательности их выполнения, завершенности отдельных этапов проекта и проекта в целом. Возможность поддержки плановых данных, фактических данных и их анализа. Типичные данные включают графики (с учетом календаря, рабочих часов, выходных и др.), компьютерные ресурсы, распределение персонала, бюджет и др.
- *оценка*. Возможность оценивать затраты, график и другие проектные параметры, вводимые пользователями.
- *управление процедурой тестирования*. Поддержка управления процедурами и программой тестирования, например, управления расписанием планируемых процедур, фиксация и запись результатов тестирования, генерация отчетов и т.д.
- *управление качеством*. Ввод соответствующих данных, их анализ и генерация отчетов.
- *корректирующие действия*. Поддержка управления корректирующими действиями, включая обработку сообщений о проблемных ситуациях.

Надежность

- администрирование репозитория. Контроль и обеспечение целостности проектных данных.
- автоматическое резервирование (определяемое поставщиком или планируемое пользователем).
- безопасность. Защита от несанкционированного доступа.
- обработка ошибок. Обнаружение ошибок в работе системы, извещение пользователя, корректное завершение работы или сохранение состояния к моменту прерывания.
- анализ отказов в критических приложениях.

Простота использования

- удобство пользовательского интерфейса. Удобство расположения и представления часто используемых элементов экрана, способов ввода данных и др.
- локализация (в соответствии с требованиями данной страны).
- простота освоения. Трудовые и временные затраты на освоение средств.
- адаптируемость к конкретным требованиям пользователя. Адаптируемость к различным алфавитам, режимам текстового и графического представления (слева-направо, сверху-вниз), различным форматам даты, способам ввода/вывода (экранным формам и форматам), изменениям в методологии (изменениям графических нотаций, правил, свойств и состава предопределенных объектов) и др.
- качество документации (полнота, понятность, удобочитаемость, полезность и др.).
- доступность и качество учебных материалов. Они могут включать компьютерные учебные материалы, учебные пособия, курсы.
- требования к уровню знаний. Квалификация и опыт, необходимые для эффективного использования CASE-средств.
- простота работы с CASE-средством (как для начинающих, так и для опытных пользователей).
- унифицированность пользовательского интерфейса (по отношению к другим средствам, использующимся в данной организации).
- онлайновые подсказки (полнота и качество).

- качество диагностики (понятность и полезность диагностических сообщений для пользователя).
- допустимое время реакции на действия пользователя (в зависимости от среды).
- простота установки и обновления версий.

Эффективность

- требования к техническим средствам. Требования к оптимальному размеру внешней и оперативной памяти, типу и производительности процессора, обеспечивающим приемлемый уровень производительности.
- эффективность рабочей нагрузки. Эффективность выполнения CASE-средством своих функций в зависимости от интенсивности работы пользователя (например, количество нажатий клавиш или кнопки мыши, требуемое для выполнения определенных функций).
- производительность. Время, затрачиваемое CASE-средством для выполнения конкретных задач (например, время ответа на запрос, время анализа 100000 строк кода). В некоторых случаях данные оценки производительности можно получить из внешних источников.

Сопровождаемость

- уровень поддержки со стороны поставщика (скорость разрешения проблем, поставки новых версий, обеспечение дополнительных возможностей).
- трассируемость обновлений (простота освоения отличий новых версий от существующих).
- совместимость обновлений (совместимость новых версий с существующими, включая, например, совместимость по входным или выходным данным).
- сопровождаемость конечного продукта (простота внесения изменений в ПО и документацию).

Переносимость

- совместимость с версиями ОС (возможность работы в среде различных версий одной и той же ОС, простота модификации CASE-средства для работы с новыми версиями ОС).
- переносимость данных между различными версиями CASE-средства.
- соответствие стандартам переносимости. Такие стандарты включают документацию, коммуникации и пользовательский интерфейс, оконный интерфейс, языки программирования, языки запросов и др.

Общие критерии

Приведенные ниже критерии являются общими по своей природе и не принадлежат к совокупности показателей качества, приведенной в стандарте ISO/IEC 9126: 1991.

- затраты на CASE-средство. Включают стоимость приобретения, установки, начального сопровождения и обучения. Следует учитывать цену для всех необходимых конфигураций (включая единственную копию, несколько копий, локальную лицензию, лицензию для предприятия, сетевую лицензию).
- оценочный эффект от внедрения CASE-средства (уровень продуктивности, качества и т.д.). Такая оценка может потребовать экономического анализа.
- профиль дистрибьютора. Общие показатели возможностей дистрибьютора. Профиль дистрибьютора может включать величину его организации, стаж в бизнесе, финансовое положение, список любых дополнительных продуктов, деловые связи (в частности, с другими дистрибьюторами данного средства), планируемая стратегия развития.
- сертификация поставщика. Сертификаты, полученные от специализированных организаций в области создания ПО (например, SEI и ISO), удостоверяющие, что квалификация поставщика в области создания и сопровождения ПО удовлетворяет некоторым минимально необходимым или вполне определенным требованиям. Сертификация может быть неформальной, например, на основе анализа качества работы поставщика.

- лицензионная политика. Доступные возможности лицензирования, право копирования (носителей и документации), любые ограничения и/или штрафные санкции за вторичное использования (подразумевается продажа пользователем CASE-средства продуктов, в состав которых входят некоторые компоненты CASE-средства, использовавшиеся при разработке продуктов).
- экспортные ограничения.
- профиль продукта. Общая информация о продукте, включая срок его существования, количество проданных копий, наличие, размер и уровень деятельности пользовательской группы, система отчетов о проблемах, программа развития продукта, совокупность применений, наличие ошибок и др.
- поддержка поставщика. Доступность, реактивность и качество услуг, предоставляемых поставщиком для пользователей CASE-средств. Такие услуги могут включать телефонную "горячую линию", местную техническую поддержку, поддержку в самой организации.
- доступность и качество обучения. Обучение может проводиться на территории поставщика, пользователя или где-либо в другом месте.
- адаптация, требуемая для внедрения CASE-средств в организации пользователя. Примером может быть определение способа использования централизованного CASE-средства с единой, общей БД в распределенной среде.

Выполнение пилотного проекта

Перед полномасштабным внедрением выбранного CASE-средства в организации выполняется пилотный проект, целью которого является экспериментальная проверка правильности решений, принятых на предыдущих этапах, и подготовка к внедрению.

Пилотный проект представляет собой первоначальное реальное использование CASE-средства в предназначенной для этого среде и обычно подразумевает более широкий масштаб использования CASE-средства по отношению к тому, который был достигнут во время оценки. Пилотный проект должен обладать многими из характеристик реальных проектов, для которых предназначено данное средство. Он преследует следующие цели:

- подтвердить достоверность результатов оценки и выбора;
- определить, действительно ли CASE-средство годится для использования в данной организации, и если да, то определить наиболее подходящую область его применения;
- собрать информацию, необходимую для разработки плана практического внедрения;
- приобрести собственный опыт использования CASE-средства.

Пилотный проект позволяет получить важную информацию, необходимую для оценки качества функционирования CASE-средства и его поддержки со стороны поставщика после того, как средство установлено.

Важной функцией пилотного проекта является принятие решения относительно приобретения или отказа от использования CASE-средства. Провал пилотного проекта позволяет избежать более значительных и дорогостоящих неудач в дальнейшем, поскольку пилотный проект обычно связан с приобретением относительно небольшого количества лицензий и обучением узкого круга специалистов.

Первоначальное использование новой CASE-технологии в пилотном проекте должно тщательно планироваться и контролироваться. Пилотный проект включает следующие шаги (рисунок 3).

Определение характеристик пилотного проекта

Пилотный проект должен обладать следующими характеристиками:

- *Область применения*. Чтобы облегчить окончательное определение области применения CASE-средства, предметная область пилотного проекта должна быть типичной для обычной деятельности организации. Пилотный проект должен помочь определить любую дополнительную технологию, обучение или поддержку, которые необходимы для перехода от пилотного проекта к широкомасштабному использованию средства. В

рамках этих ограничений пилотный проект должен иметь небольшой, но значимый размер.

- *Масштабируемость*. Результаты, полученные в пилотном проекте, должны показать масштабируемость средства. Цель - получить четкое представление о масштабах проектов, для которых данное средство применимо.

Рис. 3. Шаги пилотного проекта

- *Представительность*. Пилотный проект не должен быть необычным или уникальным для организации. CASE-средство должно использоваться для решения задач, относящихся к предметной области, хорошо понимаемой всей организацией.
- *Критичность*. Пилотный проект должен иметь существенную значимость, чтобы оказаться в центре внимания, но не должен быть критичным для успешной деятельности организации в целом. Необходимо осознавать, что первоначальное внедрение новой технологии подразумевает определенный риск. При выборе пилотного проекта приходится решать следующую дилемму: успех незначительного проекта может остаться незамеченным, с другой стороны, провал значимого проекта может вызвать чрезмерную критику.
- *Авторитетность*. Группа специалистов, участвующих в проекте, должна обладать высоким авторитетом, при этом результаты проекта будут всерьез восприняты остальными сотрудниками организации.
- *Характеристики проектной группы*. Проектная группа должна обладать готовностью к нововведениям, технической зрелостью и приемлемым уровнем опыта и знаний в данной технологии и предметной области. С другой стороны, группа должна отражать в миниатюре характеристики всей организации в целом.

В большинстве случаев существует баланс между желанием реализовать идеальный пилотный проект и реальными ограничениями организации. Организация должна выбрать пилотный проект таким образом, чтобы, во-первых, способ использования CASE-средства в нем совпадал с дальнейшими планами, и, во-вторых, перечисленные выше характеристики были сбалансированы с реальными условиями организации.

Кроме того, организация должна учитывать продолжительность пилотного проекта (и в целом процесса внедрения). Слишком продолжительный проект связан с риском потери интереса к нему со стороны руководства.

Планирование пилотного проекта

Планирование пилотного проекта должно по возможности вписываться в обычный процесс планирования проектов в организации. План должен содержать следующую информацию:

- цели, задачи и критерии оценки;

- персонал;
- процедуры и соглашения;
- обучение;
- график и ресурсы.

Цели, задачи и критерии оценки

Ожидаемые результаты пилотного проекта должны быть четко определены. Степень соответствия этим результатам представляет собой основу для последующей оценки проекта. Для определения целей, задач и критериев оценки необходимо выполнить следующие действия:

- *описать проект в терминах ожидаемых результатов* (т.е. конечного продукта). Описание должно включать форму представления и содержание результатов. Должны быть четко определены договорные требования и соответствующие стандарты.
- *определить общие цели проекта*. Примером цели может быть определение степени улучшения качества проектной документации в результате применения CASE-средств.
- *определить конкретные задачи, реализующие поставленные цели*. Каждой цели можно поставить в соответствие одну или несколько конкретных задач с количественно оцениваемыми результатами. Примером такой задачи может быть сравнительный анализ качества документации, полученной с помощью CASE-средства и без него. Документация может включать спецификацию требований к ПО, высокоуровневые и детальные проектные спецификации.
- *определить критерии оценки результатов*. Чтобы определить степень успеха пилотного проекта, необходимо использовать набор критериев, основанных на упомянутых выше задачах. Примером критерия может быть степень непротиворечивости проектной документации и контролируемости выполнения требований к ПО. Значения критериев должны сравниваться с базовыми значениями, полученными до выполнения пилотного проекта.

Персонал

Специалисты, выбранные для участия в пилотном проекте, должны иметь соответствующий авторитет и влияние и быть сторонниками новой технологии. Группа должна включать как технических специалистов, так и менеджеров, заинтересованных в новой технологии и разбирающихся в ее использовании. Группа должна обладать высокими способностями к коммуникации, знанием особенностей организационных процессов и процедур, а также предметной области. Группа не должна, тем не менее, состоять полностью из специалистов высшего звена, она должна представлять средний уровень организации.

Многие CASE-средства обеспечивают возможности, связанные с генерацией проектной документации и конфигурационным управлением. Специалисты, связанные с этими и другими смежными аспектами разработки и сопровождения ПО, также должны быть включены в состав группы.

После завершения пилотного проекта группа должна быть открыта для обмена информацией с остальными специалистами организации относительно возможностей нового средства и опыта, полученного при его использовании. Может оказаться желательным рассредоточить членов проектной группы по всей организации с целью распространения их опыта и знаний.

Процедуры и соглашения

Необходимо четко определить процедуры и соглашения, регулирующие использование CASE-средств в пилотном проекте. Эта задача скорее всего может оказаться более долгой и сложной, чем ожидается, при этом может оказаться необходимым привлечение сторонних экспертов. Примерами процедур и соглашений, которые могут повлиять на успех пилотного проекта, являются методология, технические соглашения (в частности, по наименованиям и структуре каталогов, стандарты проектирования и программирования - см. подраздел 1.3) и организационные соглашения (в частности, учет использования ресурсов, авторизация, кон-

троль изменений, процедуры экспертизы и подготовки отчетов, стандарты проверки качества).

В пилотном проекте по возможности должны использоваться принятые в организации процедуры и соглашения. С другой стороны, в течение пилотного проекта процедуры и соглашения имеют тенденцию к развитию и совершенствованию по мере накопления опыта применения средства. Существующие процедуры и соглашения могут оказаться неэффективными или чересчур ограничивающими. При этом те изменения, которые предлагается в них вносить, должны документироваться.

Обучение

Должны быть определены виды и объем обучения, необходимого для выполнения пилотного проекта. При планировании обучения нужно иметь в виду три вида потребностей: технические, управленческие и мотивационные. Ресурсы, требуемые для обучения (учебные аудитории и оборудование, преподаватели и учебные материалы), должны соответствовать плану пилотного проекта.

График обучения должен определять как специалистов, подлежащих обучению, так и виды обучения, которое они должны пройти. Обучение, которое проводится в период выполнения проекта, должно начинаться как можно быстрее после начала проекта. Обучение средствам, процессам или методам, которые не будут использоваться в течение нескольких месяцев после начала проекта, должно планироваться на то время, когда в них возникнет реальная потребность.

Поставщики CASE-средств обычно предлагают обучение использованию поставляемых ими средств. Помимо этого, для некоторых средств может быть необходимо обучение методологии. Некоторые виды обучения должны выполняться собственными силами. Такие виды обучения включают использование CASE-средства в контексте процессов, происходящих в организации, а также в совокупности с другими средствами в данной среде. Часть плана пилотного проекта, связанная с обучением, должна использоваться в качестве входа для плана практического внедрения.

При выборе необходимого обучения должны приниматься во внимание следующие факторы:

- квалификация преподавателей;
- соответствие обучения характеристикам конкретных групп специалистов (например, обзорные курсы для менеджеров, подробные курсы для разработчиков);
- возможность проведения курсов непосредственно на рабочих местах;
- возможность проведения расширенных курсов;
- возможность подготовки собственных преподавателей.

График и ресурсы

Должен быть разработан график, включающий ресурсы и сроки (этапы) проведения работ. Ресурсы включают персонал, технические средства, ПО и финансирование. Данные о персонале могут определять конкретных специалистов или требования к квалификации, необходимой для успешного выполнения пилотного проекта. Финансирование должно определяться отдельно по каждому виду работ: приобретение CASE-средств, установка, обучение, отдельные этапы проектирования.

Выполнение пилотного проекта

Пилотный проект должен выполняться в соответствии с планом. Организационная деятельность, связанная с выполнением пилотного проекта и подготовкой отчетов, должна выполняться в установленном порядке. Пилотная природа проекта требует специального внимания к вопросам приобретения, поддержки, экспертизы и обновления версий. Эти вопросы рассматриваются ниже.

Приобретение, установка и интеграция

После того, как CASE-средство выбрано, оно должно быть приобретено, интегрировано в проектную среду и настроено в соответствии с требованиями пилотного проекта. Границы этой деятельности зависят от тех действий, которые имели место в процессе оценки и выбо-

ра, а также от степени модификации средства, необходимой для его использования в проекте.

Процесс приобретения может включать подготовку контракта, переговоры, лицензирование и другую деятельность, которая выходит за рамки данных рекомендаций. Эта деятельность требует затрат времени и человеческих ресурсов, которые должны быть учтены при планировании. План должен предусматривать возможность отказа от выбранного средства на данном этапе из-за договорных разногласий.

После того, как процесс приобретения завершен, средство должно быть установлено, оттестировано и принято в эксплуатацию. Тестирование позволяет убедиться, что поставленный продукт соответствует требованиям контракта, обладает необходимой полнотой и корректностью. Этап приемки может быть предусмотрен контрактом, его реальный срок может отличаться от того, который был предусмотрен первоначально в плане пилотного проекта. Особое внимание необходимо уделить соблюдению всех требований поставщика к параметрам среды функционирования CASE-средства.

После завершения приемки может потребоваться настройка и интеграция. Настройка может включать модификацию интерфейсов, связанную с требованиями специалистов проектной группы, а также установкой прав доступа и привилегий. Настройка должна оставаться в рамках тех возможностей, которые предоставляет само средство. Не следует заниматься модификацией готовых продуктов на уровне исходных кодов.

Если новое средство должно использоваться в совокупности с некоторыми другими средствами, необходимо определить взаимодействие средств и требуемую интеграцию. Для интеграции новых средств с существующими может потребоваться построение специальных оболочек. Сложная интеграция может потребовать привлечения сторонних экспертов.

Поддержка

Доступная поддержка должна включать (по соглашению) "горячую линию" поставщика и поддержку местного поставщика, поддержку в самой организации, контакты с опытными пользователями в других организациях и участие в работе групп пользователей.

Внутренняя поддержка должна осуществляться специалистами, знакомыми с установкой средств и работой с ними. Существует несколько возможных вариантов получения такой поддержки (например, от специалиста данной организации, имеющего опыт предшествующей работы со средством; участников процесса оценки и выбора или опытного консультанта). Такой тип поддержки должен специальным образом планироваться и администрироваться. Особое внимание должно быть уделено средствам, работающим в сетях или обладающих репозиториями, поддерживающими многопользовательскую работу.

Периодические экспертизы

Обычные процедуры экспертизы проектов, существующие в организации, должны выполняться и для пилотного проекта, при этом особое внимание должно уделяться именно пилотным аспектам проекта. Помимо этого, результаты экспертиз должны служить мерой успешного использования CASE-средств.

Обновление версий

Пользователи CASE-средства могут ожидать периодического обновления версий со стороны поставщика в течение выполнения пилотного проекта. При этом необходимо тщательное отношение к интеграции этих версий. Следует заранее оценить влияние этих обновлений на ход проекта. Новые версии могут как обеспечить новые возможности, так и породить новые проблемы. Например, новая версия может потребовать видоизмененного или дополнительного обучения, а также может оказать отрицательное воздействие на уже выполненную к этому моменту работу.

Оценка пилотного проекта

После завершения пилотного проекта его результаты необходимо оценить и сопоставить их с изначальными потребностями организации, критериями успешного внедрения CASE-средств, базовыми метриками и критериями успеха пилотного проекта. Такая оценка должна установить возможные проблемы и важнейшие характеристики пилотного проекта,

которые могут повлиять на пригодность CASE-средства для организации. Она должна также указать проекты или структурные подразделения внутри организации, для которых данное средство является подходящим. Помимо этого, оценка может дать информацию относительно совершенствования процесса внедрения в дальнейшем.

В процессе оценки пилотного проекта организация должна определить свою позицию по следующим трем вопросам:

- Целесообразно ли внедрять CASE-средство?
- Какие конкретные особенности пилотного проекта привели к его успеху (или неудаче)?
- Какие проекты или подразделения в организации могли бы получить выгоду от использования средств?

Принятие решения о целесообразности внедрения CASE-средств

На данном этапе процесса внедрения организация должна сделать существенные инвестиции в CASE-средства. Если средства удовлетворили или даже превысили ожидания организации, то решение о внедрении может быть принято достаточно просто и быстро.

С другой стороны, может оказаться, что в рамках пилотного проекта средства не оправдали тех ожиданий, которые на них возлагались, или же в пилотном проекте они использовались удовлетворительно, однако опыт показал, что дальнейшие вложения в средства не гарантируют успеха.

Возможны четыре категории результатов и соответствующих действий:

- Пилотный проект потерпел неудачу, и его анализ показал неадекватность ожиданий организации. В этом случае организация может пересмотреть результаты проекта в контексте более реалистичных ожиданий.
- Пилотный проект потерпел неудачу, и его анализ показал, что выбранные средства не удовлетворяют потребности организации. В этом случае организация может принять решение не внедрять данные средства, однако при этом также пересмотреть свои потребности и подход к оценке и выбору CASE-средств.
- Пилотный проект потерпел неудачу, и его анализ показал наличие таких проблем, как неудачный выбор пилотного проекта, неадекватное обучение и недостаток ресурсов. В этом случае может оказаться достаточно сложно принять решение о том, следует ли вновь выполнить пилотный проект, продолжить работу по внедрению или отказаться от CASE-средств. Однако, независимо от принятого решения, процесс внедрения нуждается в пересмотре и повышенном внимании.
- Пилотный проект завершился успешно, и признано целесообразным внедрять CASE-средства в некоторых подразделениях или, возможно, во всей организации в целом. В этом случае следующим шагом является определение наиболее подходящего масштаба внедрения.

В ряде случаев анализ пилотного проекта может показать, что причиной неудачи явился более чем один фактор. Последующие попытки внедрения CASE-технологии должны четко выявить все причины неудачи. В экстремальных случаях тщательный анализ может показать, что в настоящий момент организация просто не готова к успешному внедрению сложных CASE-средств. В такой ситуации организация может попытаться решить свои проблемы другими средствами.

Особенности пилотного проекта

Очень важно провести анализ пилотного проекта с тем, чтобы определить его элементы, являющиеся критическими для успеха, и определить степень отражения этими элементами деятельности организации в целом. Например, если в пилотном проекте участвуют самые лучшие программисты организации, он может закончиться успешно даже вопреки использованию CASE-средств, а не благодаря им. С другой стороны, CASE-средства могут быть применены для разработки приложения, для которого они явно не подходят по своим характеристикам. Тем не менее, такое использование могло бы указать на область наиболее рационального применения средств в данной организации.

Важнейшие характеристики пилотного проекта, не являющиеся представительными для организации в целом, могут включать следующие:

- Процессы в пилотном проекте в чем-либо отличаются от процессов во всей организации.
- Квалификация группы пилотного проекта не отражает квалификацию остальных специалистов организации.
- Ресурсы, выделенные на выполнение проекта, отличаются от тех, которые выделяются для обычных проектов.
- Предметная область или масштаб проекта не соответствуют другим проектам.

Выгода от использования CASE-средств

Пилотный проект следует сопоставить с деятельностью организации в целом с тем, чтобы определить наиболее существенное сходство и отличие. Например, если наиболее заинтересованные и квалифицированные участники проекта столкнулись с серьезными трудностями в освоении средств, то менее заинтересованным и квалифицированным программистам из других подразделений потребуется существенно большее обучение.

Пилотный проект может также показать, что средства целесообразно использовать для некоторых классов проектов и нецелесообразно для других. Например, средство формальной верификации может подходить для жизненно важных приложений и не подходить для менее критических приложений.

Перед разработкой плана перехода организация должна оценить ожидаемый эффект для различных подразделений или классов проектов. При этом следует учитывать, что некоторые подразделения могут не обладать необходимой квалификацией или ресурсами для использования CASE-средств.

Принятие решения о внедрении

Возможным решением должно быть одно из следующих:

- *Внедрить средство*. В этом случае рекомендуемый масштаб внедрения должен быть определен в терминах структурных подразделений и предметной области.
- *Выполнить дополнительный пилотный проект*. Такой вариант должен рассматриваться только в том случае, если остались конкретные неразрешенные вопросы относительно внедрения CASE-средства в организации. Новый пилотный проект должен быть таким, чтобы ответить на эти вопросы.
- *Отказаться от средства*. В этом случае причины отказа от конкретного средства должны быть определены в терминах потребностей организации или критериев, которые остались неудовлетворенными. Перед тем, как продолжить деятельность по внедрению CASE-средств, потребности организации должны быть пересмотрены на предмет своей обоснованности.
- *Отказаться от использования CASE-средств вообще*. Пилотный проект может показать, что организация либо не готова к внедрению CASE-средств, либо автоматизация данного аспекта процесса создания и сопровождения ПО не дает никакого эффекта для организации. В этом случае причины отказа от CASE-средств должны быть также определены в терминах потребностей организации или критериев, которые остались неудовлетворенными. При этом необходимо понимать отличие этого варианта от предыдущего, связанного с недостатками конкретного средства.

Результатом данного этапа является документ, в котором обсуждаются результаты пилотного проекта и детализируются решения по внедрению.

Переход к практическому использованию CASE-средств

Процесс перехода к практическому использованию CASE-средств начинается с разработки и последующей реализации плана перехода. Этот план может отражать поэтапный подход к переходу, начиная с тщательно выбранного пилотного проекта до проектов с существенно возросшим разнообразием характеристик.

Разработка плана перехода

План перехода должен включать следующее:

- Информацию относительно целей, критериев оценки, графика и возможных рисков, связанных с реализацией плана.
- Информацию относительно приобретения, установки и настройки CASE-средств.
- Информацию относительно интеграции каждого средства с существующими средствами, включая как интеграцию CASE-средств друг с другом, так и их интеграцию в процессы разработки и эксплуатации ПО, существующие в организации.
- Ожидаемые потребности в обучении и ресурсы, используемые в течение и после завершения процесса перехода.
- Определение стандартных процедур использования средств.

Цели, критерии оценки, график и риски, связанные с планом перехода

Данная информация должна включать следующее:

- Типы проектов, в которых в конечном счете будет использоваться средство.
- График перехода к практическому использованию средства в отдельных проектах, который включает необходимую подготовку к его широкому использованию.
- График внедрения средства в терминах количества пользователей, включая необходимое обучение.
- Возможные риски и непредвиденные обстоятельства.
- Источники существующих (базовых) данных и метрики для оценки изменений, вызванных использованием средств.

В дополнение к сказанному, следует уделить особое внимание вопросам контроля изменений. Роли высшего руководства, субъектов и объектов изменений должны быть уточнены по сравнению с пилотным проектом, поскольку технология подлежит широкому распространению в организации.

Подразумевается, что план перехода успешно выполнен, когда не требуется больше специального планирования поддержки использования средства. В этот момент использование средства согласуется с тем, что от него ожидалось, и план работы с ним включается в общий план текущей поддержки ПО, существующий в организации.

Приобретение, установка и настройка средств

Приобретение, установка и настройка, выполненные в рамках пилотного проекта, могут потребоваться в более широком масштабе. При этом необходима следующая информация:

- Совокупность программных компонент, документации и обучения, которые необходимо приобретать для каждой отдельной платформы.
- Механизм получения новых версий.
- Настройка средства, необходимая для выполнения существующих в организации процедур и соглашений.
- Лицо или подразделение, ответственное за установку, интеграцию, настройку и эксплуатацию средства.
- План конвертирования данных и снятия старых средств с эксплуатации.

Задачи приобретения, установки и настройки должны быть как можно быстрее переданы из группы пилотного проекта в существующую службу системной поддержки ПО организации.

Интеграция средства с существующими средствами и процессами

Интеграция нового средства с существующими средствами и процессами является важным шагом в полномасштабном внедрении средства. В большинстве случаев такая интеграция в процессе пилотного проектирования не осуществляется, однако накапливаемая в этом процессе информация может помочь в разработке планов интеграции. Для планирования интеграции необходима следующая информация:

- Наименования и версии существующих средств, с которыми должно интегрироваться новое средство.
- Описания данных, которые должны совместно использоваться новым и существующими средствами, а также предварительная информация об источниках этих данных.

- Описания других взаимосвязей между новым и существующими средствами (таких, как связи по передаче управления и порядку использования), а также предварительная информация о механизмах поддержки этих взаимосвязей.
- Оценки затрат, сроков и рисков, связанных с интеграцией (и, возможно, переходом от существующих средств и данных).
- Определение способа внедрения данного средства в деятельность по совершенствованию существующих процессов.
- Ожидаемые изменения в существующих процессах и продуктах, являющиеся следствием использования нового средства и оцениваемые, по возможности, количественно.

Риск, связанный с интеграцией нового средства с существующими средствами и процессами, снижается, если потребности в интеграции учитываются в процессе оценки и выбора средства.

Обучение и ресурсы, используемые в течение и после завершения процесса перехода

Данная информация должна включать следующее:

- Персонал (включая пользователей, администраторов и интеграторов), нуждающийся в обучении использованию средства.
- Вид обучения, необходимого для каждой категории пользователей и обслуживающего персонала, с учетом особой важности обучения совместному использованию различных средств, а также методам и процессам, связанным с данными средствами.
- Вид обучения, необходимого для различных специалистов (например, для группы тестирования и независимой службы сертификации).
- Частота обучения.
- Виды и доступность поддержки.

Определение стандартов и процедур использования средств

План перехода должен определять следующие стандарты и процедуры использования средств:

- Руководства по моделированию и проектированию.
- Соглашения по присвоению имен.
- Процедуры контроля качества и процессов приемки, включая расписание экспертиз и используемые методологии.
- Процедуры резервного копирования, защиты мастер-копий и конфигурирования базы данных.
- Процедуры интеграции с существующими средствами и базами данных.
- Процедуры совместного использования данных и контроля целостности БД.
- Стандарты и процедуры обеспечения секретности.
- Стандарты документирования.

Стандарты использования CASE-средств, выработанные во время пилотного проекта, должны использоваться в качестве отправной точки для разработки более полного набора стандартов использования средств в данной организации (см. подраздел 1.3). При этом должен учитываться опыт участников пилотного проекта.

Реализация плана перехода

Реализация плана перехода требует постоянного мониторинга использования CASE-средств, обеспечения текущей поддержки, сопровождения и обновления средств по мере необходимости.

Периодические экспертизы

Достигнутые результаты должны периодически подвергаться экспертизе в соответствии с графиком, план перехода должен корректироваться при необходимости. Постоянное внимание должно уделяться степени удовлетворения потребностей организации и критериев успешного внедрения CASE-средств.

Периодические экспертизы должны продолжаться и после завершения процесса внедрения. Такие экспертизы могут анализировать метрики и другую информацию, получаемую в процессе работы с CASE-средствами, чтобы определять, насколько хорошо они продолжа-

ют выполнять требуемые функции. Такие экспертизы могут также указать на необходимость дополнительной модификации процессов.

Текущая поддержка

Текущая поддержка необходима для следующего:

- Ответов на вопросы, связанные с использованием средств.
- Передачи информации о достигнутых успехах и полученных уроках другим специалистам организации.
- Модификации и совершенствования стандартов, соглашений и процедур, связанных с использованием средства.
- Интеграции новых средств с существующими и сопровождение интегрированных средств по мере появления новых версий.
- Помощи новым сотрудникам в освоении средств и связанных с ними процедур.
- Планирования и контроля обновления версий.
- Планирования внедрения новых возможностей средств в организационные процессы.

Действия, выполняемые в процессе перехода

Для поддержки процесса перехода к практическому использованию средств желательно выполнение следующих действий:

- *Поддержка текущего обучения.* Потребность в обучении может возникать периодически вследствие появления новых версий средств или вовлечения в проект новых сотрудников.
- *Поддержка ролевых функций, связанных с процессом внедрения*. Поскольку внедрение CASE-средств приводит к изменениям в культуре организации, необходимо в процессе внедрения выделить ряд ключевых ролей (такие, как руководство высшего уровня, проектная группа и целевые группы).
- *Методики управления обновлением версий.* Эти методики могут быть связаны с обновлением версий, процедурами установки, процедурами контроля качества для оценки новых версий, процедурами обновления базы данных, конфигурацией версий и средой поддержки (другие средства, операционная система и т.д.).
- *Свободный доступ к информации*. Должны быть определены механизмы, обеспечивающие свободный доступ к информации об опыте внедрения и извлеченных из этого уроках, включая доски объявлений, информационные бюллетени, пользовательские группы, семинары и публикации.
- *Налаживание тесного рабочего взаимодействия с поставщиком*. Такое взаимодействие позволяет организации быть в курсе планов поставщика и обеспечивать оперативное удовлетворение своих требований.

Для успешного внедрения CASE-средств в организации существенно важной является последовательность в их применении. Поскольку большинство систем разрабатываются коллективно, необходимо определить характер будущего использования средств как отдельными разработчиками, так и группами. Использование стандартных процедур позволит обеспечить плавный переход между отдельными стадиями ЖЦ ПО.

Как правило, все понимают, что обучение является центральным звеном, обеспечивающим нормальное использование CASE-средств в организации. Тем не менее, довольно распространенная ошибка заключается в том, что производится начальное обучение для группы неподготовленных пользователей, а затем все ограничивается минимальным текущим обучением. Участники пилотного проекта, получившие начальное обучение, могут быть высоко квалифицированными энтузиастами новой технологии, стремящимися использовать ее во что бы то ни стало. С другой стороны, для разработчиков, которые будут участвовать в проекте в дальнейшем, может потребоваться более интенсивное и глубокое обучение, а также текущая поддержка в использовании средства.

В дополнение к этому следует отметить, что каждая категория персонала (например, администраторы средств, служба поддержки рабочих мест, интеграторы средств, служба сопровождения и разработчики приложений) нуждается в различном обучении.

Обучение не должно замыкаться только на пользователях CASE-средств, обучаться должны также те сотрудники организации, на деятельность которых так или иначе оказывает влияние использование CASE-средств.

В процессе дальнейшего использования средств в организации обучение должно стать частью процесса ориентации при найме новых сотрудников и привлечении сотрудников к проектам, в которых используются CASE-средства. Обучение должно стать неотъемлемой составной частью нормативных материалов, касающихся деятельности организации, которые предлагаются новым сотрудникам.

Одна общая ошибка, которая делается в процессе перехода, заключается в недооценке ресурсов, необходимых для поддержки постоянного использования сложных CASE-средств. Рост необходимых ресурсов вызывается тремя причинами:

- Сложностью средств,
- Частотой появления новых версий,
- Взаимодействием между средствами и внешней средой.

Сложность средств приводит к возрастанию потребностей в тщательном и продуманном обучении. Кроме того, многие CASE-средства могут использоваться только квалифицированными специалистами, умеющими сопровождать проектные базы данных и оперативно реагировать на возникающие проблемы. Высокая частота обновления версий средств может привести к возникновению нетривиальных проблем, которые зачастую упускаются из виду. Такие обновления обычно пагубно отражаются на жестких планах и графиках работы. Взаимодействие между средствами и внешней по отношению к ним средой также может иногда порождать некоторые проблемы. Имеется в виду тот факт, что хотя многие средства достигли уровня минимальной несовместимости данных между отдельными версиями, проблемы обеспечения совместимости с другими элементами внешней среды остаются в силе.

Оценка результатов перехода

Программа постоянной оценки качества и продуктивности ПО имеет важное значение для следующего:

- Определения степени совершенствования процессов,
- Упреждения возможных стратегических просчетов,
- Своевременного отказа от использования устаревшей технологии.

Чтобы определить, насколько эффективно новое CASE-средство повышает продуктивность и/или качество, организация должна опираться на некоторые базовые данные. К сожалению, лишь немногие организации в настоящее время накапливают данные для реализации программы текущей количественной оценки и усовершенствования процессов. Для доказательства эффективности CASE-средств и их возможностей улучшать продуктивность необходимы такие базовые метрические данные, как:

- Использованное время,
- Время, выделенное персонально для конкретных специалистов,
- Размер, сложность и качество ПО,
- Удобство сопровождения.

Метрическая оценка должна начинаться с реальной оценки текущего состояния среды еще до начала внедрения CASE-средств и поддерживать процедуры постоянного накопления данных.

Период времени, в течение которого выполняется количественная оценка воздействия, оказываемого внедрением CASE-средств, является весьма значимой величиной с точки зрения определения степени успешности перехода. Некоторые организации, успешно внедрившие в конечном счете CASE-средства, столкнулись с кратковременными негативными эффектами в начале процесса. Другие, успешно начав, недооценили долговременные затраты на сопровождение и обучение. Вследствие этого, наиболее приемлемый временной интервал для оценки степени успешности внедрения должен быть достаточно большим, чтобы преодолеть любые негативные эффекты на начальном этапе, а также смоделировать будущие

долговременные затраты. С другой стороны, данный интервал должен соответствовать целям организации и ожидаемым результатам.

В конечном счете, опыт, полученный при внедрении CASE-средств, может отчасти изменить цели организации и ожидания, возлагаемые на CASE-средства. Например, организация может сделать вывод, что средства целесообразно использовать для большего или меньшего круга пользователей и процессов в цикле создания и сопровождения ПО. Такие изменения в ожиданиях зачастую могут дать положительные результаты, но могут также привести к внесению соответствующих корректив в определение степени успешного внедрения CASE-средств в данной организации.

Результатом данного этапа является внедрение CASE-средств в повседневную практику организации, при этом больше не требуется какого-либо специального планирования. Кроме того, поддержка CASE-средств включается в план текущей поддержки ПО в данной организации.

Ключевые вопросы:

1. Функции, ориентированные на фазы жизненного цикла;
2. Функции CASE-средств;
3. Выполнение пилотного проекта;
4. Принятие решения о целесообразности внедрения CASE-средств;
5. Интеграция нового средства с существующими средствами и процессами.

Литература:

1 Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.
2 Советов Б.Я. Информационные технологии: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.
3 Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. -М.: ФОРУМ: ИНФРА-М, 2007.-416 с. (10 экз.)

Лекция 2. Методология IDEF0. Модели AS-IS и TO-BE.

На начальных этапах создания ИС необходимо понять, как работает организация, которую мы собираемся автоматизировать. Никто в организации не знает, как она работает в той мере подробности, которая необходима для создания ИС. Руководитель хорошо знает работу в целом, но не в состоянии вникнуть в детали работы каждого рядового сотрудника. Рядовой сотрудник хорошо знает, что творится на его рабочем месте, но плохо знает, как работают коллеги. Поэтому для описания работы предприятия необходимо построить модель. BPwin как раз и предназначен для построения такой модели - функциональной модели (или модели процессов).

Обычно сначала строится модель существующей организации работы -"AS-IS" (как есть). Анализ функциональной модели позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес-процессов и насколько глубоким изменениям подвергнется существующая структура организации бизнеса. Найденные в модели "AS-IS" недостатки можно исправить при создании модели "TO-BE" (как должно быть) - модели новой организации бизнес-процессов.

Наиболее удобным языком моделирования бизнес- процессов является IDEF0, предложенный более 20 лет назад Дугласом Россом (Ранее назывался SADT - Structured Analysis and Design Technique).

Подробно методология SADT излагается в книге Дэвида А.Марка и Клемента МакГоуэна"Методология структурного анализа и проектирования SADT", издательство Метатехнология, 1993.

Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы. Процесс моделирования какой -либо системы в IDEF0 начинается с определения контекста, т.е. наиболее абстрактно-

го уровня описания системы в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель. Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами, мы должны определить, что мы будем в дальнейшем рассматривать как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования - вопросы, на которые построенная модель должна дать ответ. IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения.

Основу методологии IDEF0 составляет графический язык описания бизнес- процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Вершина этой древовидной структуры, представляющая собой самое общее описание системы и ее взаимодействия с внешней средой, называется контекстной диаграммой. После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и так далее до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся сеансы экспертизы - эксперты предметной области указывают на соответствие реальных бизнес - процессов созданным диаграммам. Найденные несоответствия исправляются и только после прохождения экспертизы без замечаний можно приступать к следующему сеансу декомпозиции. Таким образом достигается соответствие модели реальным бизнес - процессам на любом и каждом уровне модели. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели. Работы (Activity), которые означают некие поименованные процессы, функции или задачи, изображаются в виде прямоугольников. Именем работы должен быть глагол или глагольная форма (например "Изготовление детали", "Прием заказа" и т.д.). Взаимодействие работ с внешним миром и между собой описывается в виде стрелок. Стрелки представляют собой некую информацию и именуются существительными (например, "Заготовка", "Изделие", "Заказ").

Ключевые вопросы:

1. Модель в IDEF0;
2. Синтаксис описания системы;
3. Основы методологии SADT.

Литература:

1 Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.

2 Советов Б.Я. Информационные технологии: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.

3 Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. -М.: ФОРУМ: ИНФРА-М, 2007.-416 с. (10 экз.)

Лекция 3. Разработка модели с использованием нотации IDEFX1.

Исходными данными методологии являются универсальное понятие, объем которого включает объемы всех существенных для данной предметной области понятий и набор единичных высказываний о предметной области в функциональной форме. Для полноты картины кратко изложим основные этапы методологии. На первом этапе формально определяются каждое понятие А в высказываниях вида «а = Экземпляр (А)» за исключением универсального (указывается род и видовое отличие), а также функторы отображений.

Заметим, что объемам понятий в терминах теории СЗО соответствуют классы объектов. Далее все понятия за исключением универсального делятся на группы, имеющие общее

родовое понятие и схожие по форме видовые отличия и результат представляется в форме специализаций. На следующем этапе методология предлагает построить граф классов, где вершинам сопоставлены классы объектов, а ребрам – отношения (они представлены наборами функциональных форм соответствующих отображений) и специализации.

Оставшиеся нерассмотренными на текущий момент функторы представляют «отображения, в которых объект того или иного понятия определяется на основании наличия или отсутствия у него некоторого свойства». Обратные к таким отображениям называются характеристиками и задаются в алфавитно-цифровой форме: <имя класса> (<список характеристик>).

Существует несколько модификаций ER-модели, введенных различными методологиями проектирования баз данных, в данной главе помимо «классической» модели Чена, будут анализироваться нотации Баркера, IDEF1X (Integration Definition for Information Modeling) и Information Engineering (IE).

Модель данных предметной области по сути является неким представлением реального мира, причем в различных моделях данных моделируемым объектам соответствуют различные понятия, которые, очевидно, могут быть рассмотрены совместно.

В понятиям, используемым в ОМ-моделировании, сопоставляются близкие по семантике (но в общем случае не эквивалентные) понятия из других моделей.

В ER-модели для изображения множеств сущностей используются прямоугольники: стандартные (нотация Чена, IE, IDEF1X и др.) или с закругленными углами (нотация Баркера, IDEF1X для зависимых по идентификации множеств сущностей). Прямоугольники помечаются именем сущности: надпись в большинстве нотаций располагается внутри прямоугольника или над ним (в нотации IDEF1X). Таким же образом изображаются классы в UML. ORM предлагает изображать типы объектов в виде овалов (или окружностей), внутри которых располагается имя типа. Также ORM дополняет нотацию для представления типов объектов специальными символами, которые не встречаются в других моделях, рассматриваемых в данной работе. Так, если тип объектов изображается на 10диаграмме(-ах) несколько раз, то внутри овала над именем типа объектов рисуется стрелка, указывающая на присутствие другого представления данного типа, а восклицательный знак, изображаемый справа от имени типа, означает, что объекты данного типа могут существовать в схеме, не участвуя ни в одной из связей.

Объекты и явления реального мира и отношения между ними обладают характеристиками, которым в моделировании данных принято сопоставлять атрибуты. Чен определяет атрибут как «функцию, отображающую множество сущностей или связей в множество значений или в декартово произведение множеств значений». Видимо, исходя из этого определения, в «классической» нотации для ER-диаграмм атрибуты принято изображать в виде дуг, направленных от множеств сущностей или связей к множествам значений, представленных в виде помеченных именем множества значений (т.е. домена) овалов; если атрибут связывает множество сущностей (или связей) и декартово произведение множеств значений, то используются мультиребра. Производные атрибуты рисуются при помощи пунктирных дуг, а для многозначных атрибутов на конце дуги ставится пометка «М».

Следует отметить, что более распространена нотация, изображающая в виде овалов сами атрибуты (например, такую нотацию использует Дейт), причем для отрисовки контура овалов производных атрибутов используется пунктирная линия, а для многозначных атрибутов – двойная. Если атрибут составной, то составляющие изображаются в виде отдельных эллипсов, соединенных ребрами с эллипсом составного атрибута. Имена атрибутов, составляющих первичный ключ, подчеркиваются. изображения атрибутов В IDEF1X и методе Баркера используется другой подход: имена атрибутов перечисляются в прямоугольнике, соответствующем множеству сущностей, ниже имени множества сущностей (для множеств связей атрибуты не определяются). Баркер также предлагает задавать ограничения целостности на атрибуты: «обязательные» атрибуты отмечаются справа от имени

символом «*», «необязательные» – символом «о», входящие в состав уникального идентификатора сущностей из данного множества – символом «#».

Отметим, что использование многозначных атрибутов в этих нотациях не предусмотрено.

Ключевые вопросы:

1. Модификации ER-модели;
2. Нотация Чена, IE, IDEF1X;
3. Имена атрибутов.

Литература:

1 Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.
2 Советов Б.Я. Информационные технологии: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.
3 Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. -М.: ФОРУМ: ИНФРА-М, 2007.-416 с. (10 экз.)

Лекция 4. Модель информационного пространства с помощью CASE-средств верхнего уровня BPWIN.

Технология создания информационных систем (далее - ИС) предъявляет особые требования к методикам реализации и программным инструментальным средствам, а именно: -

- Реализацию проектов по созданию ИС принято разбивать на стадии анализа (прежде Чем создавать НЕС, необходимо понять и описать бизнес-логику Предметной области), проектирования (необходимо определить модули и архитектуру будущей системы), непосредственного кодирования, тестирования и сопровождения. Известно, что исправление ошибок, допущенных на предыдущей стадии, обходится примерно в 10 раз дороже, чем на текущей; откуда следует, что наиболее критическими являются первые стадий проекта. Поэтому крайне важно иметь эффективные средства автоматизации ранних этапов реализации проекта.
- Проект по созданию сложной ИС невозможно реализовать в одиночку. Коллективная работа существенно отличается от индивидуальной, поэтому при реализации крупных проектов необходимо иметь средства координации и управления коллективом разработчиков.
- Жизненный цикл создания сложной ИС сопоставим с ожидаемым временем ее эксплуатации. Другими словами, в современных условиях компании перестраивают свои бизнес-процессы примерно раз в два года, столько же требуется (если работать по традиционной технологии) для создания ИС. Может оказаться, что к моменту сдачи ИС она уже никому не нужна, поскольку компания, ее заказавшая, вынуждена перейти На новую технологию работы. Следовательно, для создания ИС жизненно необходим инструмент, значительно (в несколько раз) уменьшающий время разработки ИС.
- Вследствие значительного жизненного цикла Может оказаться, что в процессе создания системы внешние условия Изменились. Обычно внесение изменений в проект на поздних этапах создании ИС весьма трудоемкий И дорогостоящий процесс. Поэтому для успешной реализации Крупного Проекта необходимо, что бы инструментальные средства, на которых он реализуется, были достаточно гибкими к изменяющимся требованиям.

На современном рынке средств разработки ИС достаточно много систем, в той или иной степени удовлетворяющих перечисленным требованиям. В настоящей книге рассматривается вполне конкретная технология разработки, основывающаяся на решениях фирмы Computer Associates, Которая является, по мнению автора, одной из лучших на сегодняшний день по критерию стоимость/эффективность.

Для проведения анализа и реорганизации бизнес-процессов Computer Associates предлагает CASE-средство верхнего уровня BPwin, поддерживающее методологии IDEFO (функциональная модель), IDEF3 (WorkFlow Diagram) и, DFD (DataFlow Diagram). Функциональная модель предназначена для описания существующих бизнес-процессов да предприятии (так называемая модель AS-IS) и идеального положения вещей - того, к чему нужно стремиться (модель TO-BE). Методологий IDEFO предписывает построение иерархической системы диаграмм - единичных описаний фрагментов системы. Сначала проводится описание системы в целой и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция - система разбивается на подсистемы и каждая подсистема, описывается отдельно (Диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие И так далее до достижения нужной степени подробности. После каждого сеанса декомпозиции проводится сеанс экспертизы: каждая диаграмма проверяется экспертами предметной области, представителями заказчика, людьми, непосредственно участвующими в бизнес-процессе. Такая технология создания модели позволяет построить модель, адекватную предметной области на всех уровнях абстрагирования. Если в процессе моделирования нужно осветить специфические стороны технологии предприятия, BPwin позволяет переключиться на любой ветви модели на нотацию IDEF3 или DFD и создать смешанную модель. Нотация DFD включает такие понятия, как внешняя ссылка и хранилище данных, что делает ее более удобной по сравнению с IDEFO для моделирования документооборота. Методология IDEF3 включает элемент "перекрёсток", что позволяет описать логику взаимодействия компонентов системы.

На основе модели BPwin можно построить модель данных. Для построения модели данных Computer Associates предлагает мощный и удобный инструмент -ERwin. Хотя процесс преобразования модели BPwin в модель данных плохо формализуется и поэтому полностью не автоматизирован. Computer Associates предлагает удобный инструмент для облегчения построения модели данных на основе функциональной модели - механизм двунаправленной связи BPwin - ERwin. ERwin имеет два уровня представления модели - логический и физический. На логическом уровне данные не связаны с конкретной СУБД, поэтому могут быть наглядно представлены даже для неспециалистов, Физический уровень данных - это по существу отображение системного каталога, который зависит от конкретной реализации СУБД. ERwin позволяет проводить процессы прямого и обратного проектирования БД. Это означает, что по модели данных можно сгенерировать схему БД или автоматически создать модель данных на основе информации системного каталога. Кроме того, ERwin позволяет выравнивать модель и содержимое системного каталога после редактирования того либо другого. ERwin взаимодействует с популярными средствами разработки клиентской части - PowerBuilder, Visual Basic, Delphi, что позволяет автоматически генерировать код приложения, который полностью готов к компиляции и выполнению. Для разных сред разработки реализована различная техника кодогенерации. Код для PowerBuilder генерируется непосредственно в среде ERwin, код для Visual Basic - с помощью add-in-компбнентов и библиотек, подключаемых в проект Visual Basic. ERwin не поддерживает непосредственно кодогенерацию для Delphi. Код клиентского приложения для Delphi на Основе модели данных ERwin можно сгенерировать с помощью MetaBASE - продукта фирмы gs-soft.

Создание современных ИС, основанных на широком использовании распределенных вычислений, объединении традиционных и новейших информационных технологий, требует тесного взаимодействия всех участников проекта: менеджеров, бизнес-аналитиков и системных аналитиков, администраторов БД, разработчиков. Для этого использующиеся на разных этапах и разными Специалистами средства моделирования и разработки должны быть объединены общей! системой организации совместной работы. Фирма Computer Associates, предлагает систему ModelMart -хранилище моделей, к которому открыт доступ для участников проекта создания ИС. ModelMart удовлетворяет всем требованиям, предъявляемым к средствам разработки крупных ИС, а именно:

1. Совместное моделирование. Каждый участник проекта имеет инструмент поиска и доступа к интересующей его модели в любое время-. При совместной работе используются три режима: незащищенный, защищенный и режим просмотра. В режиме просмотра запрещается любое изменение моделей. В защищенном режиме модель, с которой работает один пользователь, не может (быть изменена другими пользователями. В незащищенном режиме пользователи могут работать с общими моделями в реальном -масштабе времени. Возникающие при этом конфликты разрешаются при помощи специального модуля - Intelligent/Conflict Resolution (ICR). В дополнение к стандартным средствам организации совместной работы ModelMart позволяет сохранять множество версий, снабженных аннотациями, с последующим сравнением предыдущих и новых версий. При необходимости возможен возврат к предыдущим версиям.
2. Создание библиотек решений. ModelMart позволяет формировать библиотеки стандартных решений, включающие наиболее удачные фрагменты реализованных проектов, накапливать и использовать типовые модели, объединяя их при необходимости "сборки" больших систем. На основе существующих БД с помощью ERwin возможно восстановление моделей (обратное проектирование), которые в процессе анализа пригодности их для новой системы могут объединяться с типовыми моделями из библиотек моделей.
3. Управление доступом. Для каждого участника проекта определяются права доступа, в соответствии с которыми они получают возможность работать только с определенными моделями. Права доступа могут быть определены как для групп, так и для отдельных участников проекта. Роль специалистов, участвующих в различных проектах" может меняться, поэтому в ModelMart можно определять права доступа и управлять правами доступа участников проекта к библиотекам, моделям и даже к специфическим областям модели.
4. Архитектура ModelMart. Система ModelMart реализована на архитектуре клиент-сервер. В качестве платформы реализации хранилища выбраны РСУБД Sybase, Microsoft S,QL Server, Informix и Oracle. Клиентскими приложениями являются ERwin 3.x и BPwin 2.x. В ModelMart реализован доступ к хранилищу моделей через API, что позволяет постоянно наращивать возможности интегрированной среды путем включения новых инструментов моделирования и анализа,

Как было указано выше (см. п. С), при разработке крупных проектов критическим становится время реализации проекта. Одним из решений Проблемы может стать автоматическая генерация кода приложения (клиентской части) CASE-средствами на основе модели предметной области. Хотя ERwin решает эту задачу, код генерируется на основе модели IDEF1X, т. е. фактически на основе реляционной модели данных. Которая непосредственно не содержит информации о бизнес-процессах. Как следствие этого сгенерированный код не может полностью обеспечить функциональность приложения со сложной бизнес-логикой. Объектно-ориентированное проектирование - альтернативная технология кодогенерации, которая лишена этого недостатка.

Существует несколько CASE-средств, поддерживающих языки объектно-ориентированного проектирования, в том числе ставший в последнее время стандартом UML. Наиболее известными являются Paradigm Plus фирмы Computer Associates и выпущенный фирмой Rational Software программный пакет Rational Rose. Эти инструменты позволяют строить объектные модели в различных нотациях (OMT, UML, Буч и др.) и генерировать на основе полученной модели приложения на языках программирования C++, Visual Basic, Power Builder, Java, Ada, Smalltalk и др. Поскольку генерация кода реализована на основе знаний предметной области, а не на основе реляционной структуры данных, полученный код более полно отражает бизнес-логику. Rational Rose- и Paradigm Plus поддерживают не только прямую генерацию, кода, но и обратное проектирование, т. е. создание объектной модели по исходному коду приложения.

Ключевые вопросы:

1. Технология создания информационных систем;

2. CASE-средство верхнего уровня BPwin;
3. Модель данных;
4. Требования к средствам разработки ИС;
5. CASE-средства, поддерживающие языки объектно-ориентированного проектирования.

Литература:

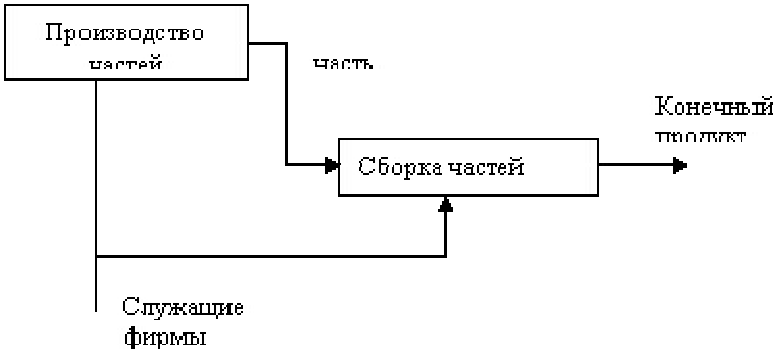
1 Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.
2 Советов Б.Я. Информационные технологии: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.
3 Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. -М.: ФОРУМ: ИНФРА-М, 2007.-416 с. (10 экз.)

Лекция 5. Модель информационного пространства с помощью CASE-средств верхнего уровня ERWIN.

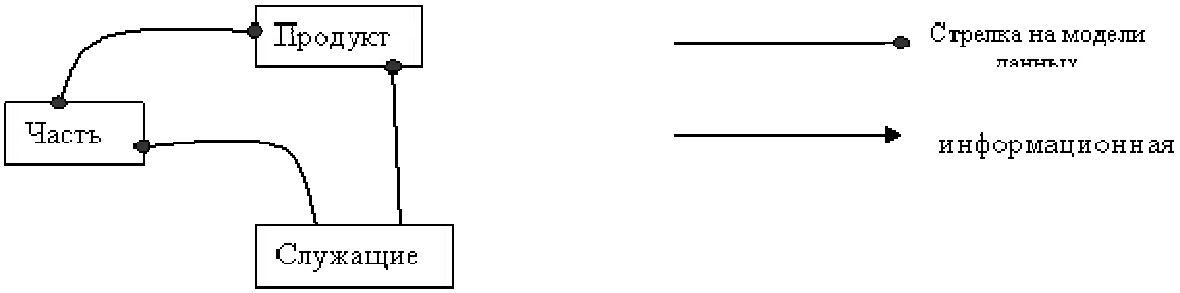
Модель ERWin поддерживает два уровня представления модели данных: логический уровень и физический уровень. Логический уровень не зависит от конкретной реализации БД и позволяет наглядно представить данные для обсуждения с экспертами БД. Физический уровень является отображением системного каталога БД и зависит от конкретной реализации БД.

На логическом уровне модели данных информация изображается в виде сущности.

Сущности соответствуют таблицам на физическом уровне БД. Сущности состоят из атрибутов. Атрибуты соответствуют колонкам таблицы. Сущности состоят из совокупности отдельных данных, которые называются экземплярами сущностей. Экземпляр сущности соответствует записи в таблице. К модели данных представляются определенные требования, которые называются нормализацией данных. Нормализация проводится для обеспечения компактности и непротиворечивости данных. Основная идея нормализации данных заключается в том, что факт действует в одном месте. Это приводит к тому, что информация, которая моделируется одной стрелкой в модели процессов, может содержаться в нескольких сущностях и атрибутах в модели данных, кроме того, на модели процессов могут присутствовать стрелки, отображающие одни и те же данные, но на разных этапах обработки (не обработанная и обработанная деталь, и т.п.).



1. Стрелке в модели процессов может соответствовать отдельная сущность модели данных.



2. Информация о стрелке может содержаться только в нескольких атрибутах одной сущности. Различным атрибутам одной сущности могут соответствовать различные стрелки.
3. Работы в модели процессов могут создавать или изменять данные, которые соответствуют входным и выходным структурам, они могут воздействовать как целиком на сущности, создавая или модифицируя экземпляр сущности, так и на отдельные атрибуты сущности.

Модель BPWin позволяет связывать модели элементов данных, создающихся с помощью ERWin. Построение моделей данных представляет определение сущностей и их атрибутов, то есть надо знать какая информация будет храниться в каждой сущности и в атрибутах. Атрибутов может быть много. Каждый атрибут хранит информацию об определенном свойстве сущности. Каждый экземпляр сущности должен быть уникальным, атрибут или группа атрибутов, которые идентифицируют сущность, называются первичным ключом. При определении первичного ключа может быть рассмотрено несколько наборов атрибутов. К первичным ключам представляются определенные требования: Первичный ключ должен однозначно идентифицировать экземпляр сущности. Первичный ключ должен быть компактен, то есть удаление любого атрибута из составного первичного ключа приводит к потере уникальности экземпляра сущности. Каждый атрибут из состава первичного ключа не должен принимать нулевого значения. Каждый атрибут первичного ключа не должен менять своего значения в течении всего времени существования экземпляра сущности.

К модели данных представляются требования называемые нормальными формами. *1 нормальная форма* требует, чтобы все атрибуты были атомарными, не должно быть атрибута типа «адрес», а должен быть «индекс», «страна», «улица» и т.д. *2 нормальная форма* требует, чтобы не ключевые атрибуты зависели от всего первичного, а не от его части. После завершения проектирования модель данных может быть перенесена в целевую среду СУБД сервера.

Ключевые вопросы:
1. Модель ERWin;
2. Требования к модели;
3. Построение моделей;

Литература:
1 Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.
2 Советов Б.Я. Информационные технологии: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.
3 Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. -М.: ФОРУМ: ИНФРА-М, 2007.-416 с. (10 экз.)

Лекция 6. Декомпозиция. Отношение между классами. Иерархия классов. Объекты и классы. Абстрагирование и обобщение.

Объектно-ориентированное программирование - это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

В данном определении можно выделить три части: 1) ООР использует в качестве базовых элементов *объекты,* а не алгоритмы (иерархия "быть частью", которая была определена в главе 1); 2) каждый объект является *экземпляром* какого-либо определенного *класса;* 3) классы организованы *иерархически* (см. понятие об иерархии "is a" там же). Программа будет объектно-ориентированной только при соблюдении всех трех указанных требований. В частности, программирование, не основанное на иерархических отношениях, не относится к ООР, а называется*программированием на основе абстрактных типов данных.*

В соответствии с этим определением не все языки программирования являются объектно-ориентированными. Страуструп определил так: "если термин *объектно-ориентированный язык* вообще что-либо означает, то он должен означать язык, имеющий средства хорошей поддержки объектно-ориентированного стиля программирования... Обеспечение такого стиля в свою очередь означает, что в языке удобно пользоваться этим стилем. Если написание программ в стиле ООР требует специальных усилий или оно невозможно совсем, то этот язык не отвечает требованиям ООР" [<="" p="" style="text-decoration: initial; color: rgb(0, 0, 153);">

<="" p="" style="text-decoration: initial; color: rgb(0, 0, 153);">

- Поддерживаются объекты, то есть абстракции данных, имеющие интерфейс в виде именованных операций и собственные данные, с ограничением доступа к ним.
- Объекты относятся к соответствующим типам (классам).

- <="" p="" style="text-decoration: initial; color: rgb(0, 0, 153);">Типы (классы) могут наследовать атрибуты супертипов (суперклассов)" [34].

Поддержка наследования в таких языках означает возможность установления отношения "is-a" ("есть", "это есть", " - это"), например, красная роза - это цветок, а цветок - это растение. Языки, не имеющие таких механизмов, нельзя отнести к объектно-ориентированным. Карделли и Вегнер назвали такие языки *объектными,* но не *объектно-ориентированными.* Согласно этому определению объектно-ориентированными языками являются Smalltalk, Object Pascal, C++ и CLOS, а Ada - объектный язык. Но, поскольку объекты и классы являются элементами обеих групп языков, желательно использовать и в тех, и в других методы объектно-ориентированного проектирования.

Объектно-ориентированное проектирование. Программирование прежде всего подразумевает правильное и эффективное использование механизмов конкретных языков программирования. Проектирование, напротив, основное внимание уделяет правильному и эффективному структурированию сложных систем. Мы определяем объектно-ориентированное проектирование следующим образом:

Объектно-ориентированное проектирование - это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы.

В данном определении содержатся две важные части: объектно-ориентированное проектирование 1) основывается на объектно-ориентированной декомпозиции; 2) использует многообразие приемов представления моделей, отражающих логическую (классы и объекты) и физическую (модули и процессы) структуру системы, а также ее статические и динамические аспекты.

Именно объектно-ориентированная декомпозиция отличает объектно-ориентированное проектирование от структурного; в первом случае логическая структура системы отражается абстракциями в виде классов и объектов, во втором - алгоритмами. Иногда мы будем использовать аббревиатуру *OOD,* object-oriented design, для обозначения метода объектно-ориентированного проектирования, изложенного в этой книге.

Объектно-ориентированный анализ. На объектную модель повлияла более ранняя модель жизненного цикла программного обеспечения. Традиционная техника структурного анализа, описанная в работах Де Марко [39], основана на потоках данных в системе. Объект-

но-ориентированный анализ (или *OOA, object-oriented analysis*) направлен на создание моделей реальной действительности на основе объектно-ориентированного мировоззрения.

Объектно-ориентированный анализ - это методология, при которой требования к системе воспринимаются с точки зрения классов и объектов, выявленных в предметной области.

Как соотносятся OOA, OOD и OOP? На результатах OOA формируются модели, на которых основывается OOD; OOD в свою очередь создает фундамент для окончательной реализации системы с использованием методологии OOP.

Парадигмы программирования

Дженкинс и Глазго считают, что "в большинстве своем программисты используют в работе один язык программирования и следуют одному стилю. Они программируют в парадигме, навязанной используемым ими языком. Часто они оставляют в стороне альтернативные подходы к цели, а, следовательно, им трудно увидеть преимущества стиля, более соответствующего решаемой задаче". Эти же авторы выявили пять основных разновидностей стилей программирования, которые перечислены ниже вместе с присущими им видами абстракций:

процедурно-ориентированный	алгоритмы
объектно-ориентированный	классы и объекты
логико-ориентированный	цели, часто выраженные в терминах исчисления предикатов
ориентированный на правила	правила "если-то"
ориентированный на ограничения	инвариантные соотношения

Невозможно признать какой-либо стиль программирования наилучшим во всех областях практического применения. Например, для проектирования баз знаний более пригоден стиль, ориентированный на правила, а для вычислительных задач - процедурно-ориентированный. По нашему опыту объектно-ориентированный стиль является наиболее приемлемым для широчайшего круга приложений; действительно, эта парадигма часто служит архитектурным фундаментом, на котором мы основываем другие парадигмы.

Каждый стиль программирования имеет свою концептуальную базу. Каждый стиль требует своего умонастроения и способа восприятия решаемой задачи. Для объектно-ориентированного стиля концептуальная база - это *объектная модель*. Она имеет четыре главных элемента:

- абстрагирование;
- инкапсуляция;
- модульность;
- иерархия.

Эти элементы являются *главными* в том смысле, что без любого из них модель не будет объектно-ориентированной. Кроме главных, имеются еще три дополнительных элемента:

- типизация;
- параллелизм;
- сохраняемость.

Называя их *дополнительными,* мы имеем в виду, что они полезны в объектной модели, но не обязательны.

Без такой концептуальной основы вы можете программировать на языке типа Smalltalk, Object Pascal, C++, CLOS, Eiffel или Ada, но из-под внешней красоты будет выглядывать стиль FORTRAN, Pascal или C. Выразительная способность объектно-ориентированного языка будет либо потеряна, либо искажена. Но еще более существенно, что при этом будет мало шансов справиться со сложностью решаемых задач.

Смысл абстрагирования. Абстрагирование является одним из основных методов, используемых для решения сложных задач. Хоар считает, что "абстрагирование проявляется в нахождении сходств между определенными объектами, ситуациями или процессами реального мира, и в принятии решений на основе этих сходств, отвлекаясь на время от имеющихся различий". Суммируя эти разные точки зрения, получим следующее определение абстракции:

Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя.

Абстрагирование концентрирует внимание на внешних особенностях объекта и позволяет отделить самые существенные особенности поведения от несущественных. Абельсон и Суссман назвали такое разделение смысла и реализации *барьером абстракции*. Мы считаем полезным еще один дополнительный принцип, называемый *принципом наименьшего удивления,* согласно которому абстракция должна охватывать все поведение объекта, но не больше и не меньше, и не привносить сюрпризов или побочных эффектов, лежащих вне ее сферы применимости.

Выбор правильного набора абстракций для заданной предметной области представляет собой главную задачу объектно-ориентированного проектирования.

По мнению Сейдвица и Старка "существует целый спектр абстракций, начиная с объектов, которые почти точно соответствуют реалиям предметной области, и кончая объектами, не имеющими право на существование". Вот эти абстракции, начиная от наиболее полезных к наименее полезным:

Абстракция сущности	Объект представляет собой полезную модель некой сущности в предметной области
Абстракция поведения	Объект состоит из обобщенного множества операций
Абстракция виртуальной машины	Объект группирует операции, которые либо вместе используются более высоким уровнем управления, либо сами используют некоторый набор операций более низкого уровня
Произвольная абстракция	Объект включает в себя набор операций, не имеющих друг с другом ничего общего

Мы стараемся строить абстракции сущности, так как они прямо соответствуют сущностям предметной области.

Клиентом называется любой объект, использующий ресурсы другого объекта (называемого *сервером*). Мы будем характеризовать поведение объекта услугами, которые он оказывает другим объектам, и операциями, которые он выполняет над другими объектами. Такой подход концентрирует внимание на внешних проявлениях объекта и приводит к идее, которую Мейер назвал *контрактной моделью* программирования.

Каждая операция, предусмотренная этим контрактом, однозначно определяется ее формальными параметрами и типом возвращаемого значения. Полный набор операций, которые клиент может осуществлять над другим объектом, вместе с правильным порядком, в котором эти операции вызываются, называется *протоколом.* Протокол отражает все возможные способы, которыми объект может действовать или подвергаться воздействию, полностью определяя тем самым внешнее поведение абстракции со статической и динамической точек зрения.

Ключевые вопросы:

1. Объектно-ориентированное программирование;
2. Объектно-ориентированный анализ;
3. Парадигмы программирования;
4. Смысл абстрагирования;
5. Контрактная модель.

Литература:

1 Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.
2 Советов Б.Я. Информационные технологии: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.
3 Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. -М.: ФОРУМ: ИНФРА-М, 2007.-416 с. (10 экз.)

Лекция 7. ООП. Ограниченность доступа. Модульность.

Чтобы обеспечить расширяемость (extendibility) и повторное использование (reusability), двух основных факторов качества, предложенных в лекции 1, необходима система с гибкой архитектурой, состоящая из автономных программных компонент. Именно поэтому в лекции 1 введен термин модульность (modularity), сочетающий оба фактора.

Модульное программирование ранее понималось как сборка программ из небольших частей, обычно подпрограмм. Но такой подход не может обеспечить реальную расширяемость и повторное использование программного продукта, если не гарантировать, что элементы сборки - модули - являются самодостаточными и образуют устойчивые структуры. Любое достаточно полное определение модульности должно обеспечивать реализацию этих свойств.

Таким образом, метод проектирования программного продукта является модульным, если он помогает проектировщикам создать систему, состоящую из автономных элементов с простыми и согласованными структурными связями между ними. Цель этой лекции - детализация этого неформального определения и выяснение того, какими конкретно свойствами должен обладать метод, заслуживающий название "модульного". Наше внимание будет сосредоточено на этапе проектирования, но все идеи применимы и к ранним этапам - анализа и спецификации, также как и к этапам разработки и сопровождения.

Рассмотрим модульность с разных точек зрения. Введем набор дополнительных свойств: пять критериев (criteria), пять правил (rules) и пять принципов (principles) модульности, обеспечивающих при их совместном использовании выполнение наиболее важных требований, предъявляемых к методу модульного проектирования.

Для практикующего разработчика ПО принципы и правила не менее важны, чем критерии. Различие лишь в причинной связи: критерии являются взаимно независимыми (метод может удовлетворять одному из них и в тоже время противоречить оставшимся), в то время как правила следуют из критериев, а принципы следуют из правил.

Можно было бы ожидать, что эта лекция начнется с подробного описания того, как выглядит модуль. Но это не так, и для этого есть серьезные основания. Задача этой и двух следующих лекций - анализ свойств, которыми должна обладать надлежащим образом спроектированная модульная структура. Вопросом о виде модулей мы займемся в конце нашего обсуждения, а не в его начале. И пока мы не дойдем до этой точки, слово "модуль" будет означать компонент разбиения рассматриваемой системы. Если вы знакомы с не ОО-методами, то, вероятно, вспомните о подпрограммах, имеющихся в большинстве языков программирования и проектирования, или, быть может, о пакетах (packages) языка Ada и (правда, под другим названием) языка Modula. Наконец, в последующих лекциях наше обсуждение приведет к ОО-виду модуля - классу. Даже если вы уже знакомы с классами и ОО-методами, все же следует прочитать эту лекцию для понимания требований, предъявляемых к классам, - это поможет правильному их конструированию.

Метод проектирования, который можно называть "модульным", должен удовлетворять пяти основным требованиям:

[x]. Декомпозиции (decomposability).
[x]. Композиции (composability).
[x]. Понятности (understandability).

[x]. Непрерывности (continuity).

[x]. Защищенности (protection).

Метод удовлетворяет критерию Модульной Композиции, если он обеспечивает разработку элементов программного продукта, свободно объединяемых между собой для получения новых систем, быть может, в среде, отличающейся от той, для которой эти элементы первоначально разрабатывались.

Композиция определяет процесс, обратный декомпозиции: элементы программного продукта извлекаются из того контекста, для которого они были первоначально предназначены, для использования их вновь в ином контексте.

Метод удовлетворяет критерию Модульной Понятности, если он помогает получить такую программу, читая которую можно понять содержание каждого модуля, не зная текста остальных, или, в худшем случае, ознакомившись лишь с некоторыми из них.

Важность этого критерия следует из его влияния на процесс сопровождения программного продукта. Почти все действия по сопровождению программы, как неизбежные, так и не столь неизбежные, связаны с глубоким пониманием ее элементов. Метод едва ли может называться модульным, если тот, кто читает программный текст, не в состоянии понять его смысл.

Метод удовлетворяет критерию Модульной Непрерывности, если незначительное изменение спецификаций разработанной системы приведет к изменению одного или небольшого числа модулей.

Этот критерий непосредственно связан с критерием расширяемости. Как подчеркивалось в предыдущей лекции, внесение изменений является неотъемлемой частью процесса разработки программного продукта. Соответствующие требования к программе будут неминуемо изменяться в ходе разработки. Непрерывность означает, что небольшие изменения будут воздействовать только на отдельные модули в структуре системы, а не на всю систему.

Метод удовлетворяет критерию Модульной Защищенности, если он приводит к архитектуре системы, в которой аварийная ситуация, возникшая во время выполнения модуля, ограничится только этим модулем, или, в худшем случае, распространится лишь на несколько соседних модулей.

Вопрос об отказах и ошибках является основным в программной инженерии. Сейчас речь идет об ошибках периода исполнения программы, связанных с аппаратными прерываниями, ошибочными входными данными или исчерпанием необходимых ресурсов (например, из-за недостаточного объема памяти). Критерий защищенности направлен не на предотвращение или исправление ошибок, а на проблему, непосредственно связанную с модульностью - распространением ошибок в модульной системе.

Ключевые вопросы:

1. Расширяемость;
2. Модульное программирование;
3. Критерий Модульной Непрерывности.

Литература:

1 Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.

2 Советов Б.Я. Информационные технологии: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.

3 Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. -М.: ФОРУМ: ИНФРА-М, 2007.-416 с. (10 экз.)

Лекция 8. Пакеты и классы. Уточнение методов и свойств классов.

Моделирование на языке UML является объектно-ориентированным, поэтому основной сущностью в UML является класс.

На диаграммах класс обозначается прямоугольником, в котором записывается его имя. Если диаграмма подробная, то также указываются атрибуты (поля) класса и его методы. В нашем случае класс Man (человек) имеет три поля и два метода. Поля и методы описываются в стиле «Паскаль»: сначала идет имя поля или метода, потом его тип или тип возвращаемого значения. Для методов в скобках указываются параметры, если они есть. В начале описания может стоять модификатор доступа « + », который обозначает, что член класса является публичным, то есть доступным для использования всем. Диезом «#» обозначают члены класса, доступные только самому классу и его потомкам. «-» служит для обозначения закрытых членов, доступных только самому классу.

Более крупной сущностью (по сравнению с классом) является компонент, который обычно представляет собой физическую часть системы, например DLL (динамически подключаемую библиотеку).

Аналогичную функцию выполняют пакеты: они также служат контейнерами для других сущностей, но являются более универсальным механизмом и могут служить просто для логического разделения сущностей. Например, чтобы нагляднее представить для себя структуру программы, ты объединил в пакет элементы графического интерфейса пользователя, но в реальности они будут находиться в разных компонентах. Пакеты очень удобны при проектировании «сверху вниз» или при нисходящем проектировании: при таком подходе ты разбиваешь будущую программу на большие части (пакеты), а затем детализируешь их. На диаграммах пакет обозначается в виде папки.

Отношение — это связь между сущностями определенного вида. В языке UML существует три вида основных отношений. Пожалуй, самым общим их видом является отношение зависимости (Dependency), которое указывает, что изменение одной сущности повлияет на другую сущность. В нашем случае изменения в классе Engine (движок) могут повлиять на класс Car (машина).

Вторым важным отношением является ассоциация (Association), которая показывает, что объекты одной сущности связаны с объектами другой сущности. Следует отметить, что отношение ассоциации является довольно общим понятием и существует его частный вид — композиция. На диаграммах она обозначается в виде стрелки с ромбиком (схема «отношение композиции»).

На очереди третье отношение — обобщение (Generalization). Это отношение отражает понятие объектно-ориентированного программирования «наследование». При таком отношении родительский класс обобщает дочерний, наследующий все его открытые и защищенные поля и методы.

Ключевые вопросы:

1. Основная сущность в UML;
2. Отношение;
3. Ассоциация;
4. Обобщение.

Литература:

1 Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.
2 Советов Б.Я. Информационные технологии: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.
3 Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. -М.: ФОРУМ: ИНФРА-М, 2007.-416 с. (10 экз.)

Лекция 9. Кодогенерация проекта в Delphi.

Методология, которой мы следуем при разработке ПО, это методология разработки программного обеспечения *Rational Unified Process* фирмы *Rational Software Corporation*. Методология с технологической точки зрения включает в себя следующие этапы (разумеет-

ся, в рамках каждой итерации): моделирование предметной области, определение требований к системе, анализ и проектирование, реализацию (кодирование и автономную отладку), тестирование и внедрение. Ниже приведена таблица, в которой представлено то, что мы ожидаем получить при помощи *Rational Rose* и *Delphi* на каждом этапе (см. Табл. 1.).

Табл. Использование *Rational Rose* и *Delphi* на различных этапах разработки ПО

Этап	*Что мы ожидаем от Rose*	*Что мы ожидаем от Delphi*
Моделирование предметной области	Выполняется моделирование предметной области, описывается предметная область “как есть”	Ничего
Определение требований к системе	Определяются функциональные требования к системе, требования к интерфейсу системы.	Создается прототип пользовательского интерфейса
Анализа и проектирование	Определяются базовые компоненты архитектуры, моделируются данные, детально проектируются компоненты системы.	Элементы объектной модели *Delphi* включаются в базовую архитектуру. Обеспечивается однозначное соответствие элементов диаграмм классов *Rose* и элементов компонент *Delphi*.
Реализация	Реализуются в виде программных модулей диаграммы классов, разрабатываются диаграммы компонентов и диаграммы размещения.	Реализуется программный код, обеспечивается однозначное соответствие проекта в *Rose* и *Delphi*. Документирование кода в *Delphi* отражается в *Rose*.
Тестирование	Модели остаются практически неизменными. Разрабатываются тестовые примеры для тестирования функций системы.	Вносятся изменения в программный код. Изменения в программном коде отражаются в коде *Delphi*.
Внедрение	Диаграммы размещения являются основой для внедрения ПО. На основе моделей может быть получена актуальная документация системного уровня.	Ничего.

Как можно заметить из таблицы 1, основное взаимодействие *Rose* и *Delphi* может происходить на этапах определения требований к системе, анализа и проектирования и реализации. Основными моделями, используемыми на этих этапах, являются модель функций системы, модель интерфейса, модель данных, модель спецификаций программных модулей.

Для обеспечения связи между *Delphi* и *Rose* используется промежуточное ПО называемое кодогенератором. Строго говоря, задачи этого ПО не ограничиваются только генерацией кода. Прежде, чем перейти к описанию конкретного кодогенератора перечислим, что, по нашему мнению он должен обеспечивать.

Итак, кодогенератор должен: иметь возможность преобразовывать классы *Rational Rose* в код определения классов на целевом языке, в данном случае *Delphi*, при этом описание, связанное с конкретным классом должно помещаться в соответствующее место программного кода; для диаграммы классов поддерживать стереотипы, связанные со специфическими особенностями языка (например, стереотип *unit*, *interface* или *property*, соответственно определяя, что конкретный пакет – есть модуль *Delphi* или класс - есть интерфейс *Delphi*, или атрибут - есть свойство для компоненты *Delphi*); иметь описанный, очевидный и однозначный способ отображения диаграммы классов в код *Delphi*, при этом отобра-

жение должно быть настраиваемым; иметь возможность импорта актуальной объектной модели *Delphi* (желательно для различных версий библиотеки *VCL*); поддерживать генерацию кода для создания компонент *Delphi*; уметь правильным образом отображать типовые виды связей между классами (хотя бы обобщение, агрегацию и композицию); уметь отображать в программный код кардинальность связи; исходя из диаграммы компонент *Rose*, создавать проект *Delphi*, содержащий требуемые программные модули (*forward engeneering*); на основе готового проекта *Delphi* строить диаграмму компонент *Rose*, содержащую в виде компонент все модули проекта *Delphi* и связанные с ними классы, полученные из *Delphi* в результате обратного проектирования (*reverse engeneering*). Диаграммы должны быть компактными и очевидными; обеспечивать автоматическое согласование модели *Rose* и *Delphi* после внесения изменений в код модулей *Delphi (round trip engeneering)*; после внесения изменений в модель *Rose* и повторной генерации кода не уничтожать, фрагменты, написанные в среде *Delphi*; помимо генерации кода иметь способ отображения такого важного элемента *Delphi*, как формы; на реальных проектах (сотни классов и модулей) работать с приемлемой производительностью.

Компания *Ensemble Systems, Inc.* в настоящее время является одним из ведущих поставщиков дополнительных компонент *(add ins)* для *Rational Rose.* Эти компоненты поддерживают кодогенерацию для широкого спектра популярных систем программирования, в том числе и для *Delphi*. Рассмотрим работу кодогенератора *Delphi* в среде *Rational Rose*.

После установки компонента *Delphi* от фирмы *Ensemble Systems*, *Inc*. в среде *Rational Rose* появляется новый пункт меню в разделе *Tools*.

Кодогенератор носит название *Rose Delphi Link* (*RDL*). Как пользоваться *RDL* мы опишем в следующем разделе, когда будем рассматривать конкретный пример. Сейчас же мы сосредоточим наше внимание на основных возможностях *RDL* (не имея о них представления вряд ли можно использовать его эффективно).

Прежде всего, заметим, что код, создаваемый RDL *не содержит реализацию* (для объектной модели это, как правило, это тело метода). Когда модель обновляется из кода *Delphi* (*reverse engeneering* или *round trip*), модель не подгружает программный код, написанный в среде *Delphi* для тел методов. Изменения в модели касаются только декларативных элементов: определений классов, интерфейсов, типов, записей и т. п. Однако, при повторной генерации кода из *Rose*, тела методов *в* Delphi также остаются неизменными, а меняются только декларативные элементы. Таким образом “испортить” программный код при повторной генерации нельзя. Для отображения элементов модели в программный код *RDL* использует *Code Generation Properties* (*CGP*) – набор специальных таблиц, которые связываются с каждым элементом моделей *Rational Rose* и содержат специфическую для *Delphi* информацию, используемую для кодогенерации. Набор этих таблиц доступен из главного меню (пункт *Tools*/*Options*, закладка *Delphi*).

Для того, что бы *CGP для Delphi* было доступно из спецификации необходимо установить значение поля *Default Language = Delphi* в закладке *Notation* пункта меню*Tools*/*Options*. Для упрощения работы в разделе меню *Tools*/*Ensemble Tools* появляется также закладка *Delphi Property Editor*, где можно настроить свойства выбранного элемента модели. Заметим, что использование *CGP* является типичным приемом для всех кодогенераторов от *Ensemble Systems*, Inc и для Rational Rose вообще. При использовании *Delphi Property Editor* можно, не выполняя кодогенерации, просмотреть код соответствующего элемента модели (например, класса), что часто бывает очень удобно.

Теперь рассмотрим, как отображаются *RDL* в программный код конкретные типовые приемы проектирования

1. Наследование от класса и реализация интерфейса

Преобразуется в код: type SampleClass1 = class (BaseClass, SampleInterface) {...} end;

2. Ассоциативные связи с ролями и различной кардинальностью связи

Преобразуется в код:

type SampleClass3 = class private ArrayRole1 : array of SupplierClass2; ArrayRole2 : array [1..10] of SupplierClass5; ArrayRole3 : array [SampleRange] of SupplierClass3; ArrayRole4 : TItems; end;

Заметим, что для представления ролей с кардинальностью связи не равной 1 используются массивы (либо фиксированной длины, либо динамические). Константа SampleRange в примере должна быть определена в свойстве Array_Range для роли ArrayRole. Обратите внимание, что для *Role*4 в коде определен не массив, а тип TItems, который должен представлять коллекцию.

3. Агрегация и композиция

Преобразуется в код:

SampleClass1 = class Public A : SampleClass2; B : SampleClass3; C : array of SampleClass4; D : array of SampleClass5; end;

Заметим, что мы не нашли существенной разницы в коде для агрегации и композиции и, вообще, получившийся код очень напоминает код для обычной ассоциативной связи с ролями.

4. Стереотипы для типовых элементов программного кода Delphi RDL имеет несколько стереотипов для представления типовых элементов программного кода *Delphi*: указателей, массивов, файлов, ссылок на класс и т.п. Правда, на наш взгляд, эти элементы малоупотребительны при высокоуровневом проектировании объектной модели. В заключение краткого описания особенностей *RDL* отметим следующее. Важным элементом проектирования является документация в спецификации для соответствующего элемента модели. *RDL* обеспечивает ее передачу в программный код. Так для операций (методов класса *Delphi*) значение поля *documentation* в спецификации вставляется в код перед определением метода. И обратно, комментарии в программном коде *Delphi* могут быть преобразованы в поля *documentation* в спецификации элементов модели *Rose*.

Ключевые вопросы:

1. Методология разработки программного обеспечения *Rational Unified Process;*
2. Промежуточное ПО – кодогенератор;
3. Работа кодогенератора *Delphi* в среде *Rational Rose*;
4. Типовые приемы проектирования.

Литература:

1. Гвоздева Т.В.Проектирование информационных систем: учеб. пособие: рек. УМО/ Т.В. Гвоздева, Б.А. Баллод. -Ростов н/Д: Феникс, 2009.-509 с.

2. Советов Б.Я. Информационные **технологи**и: учеб.: доп. Мин. обр. РФ/ Б.Я. Советов, В.В. Цехановский. -4-е изд., стер.. -М.: Высш. шк., 2008.-264 с.

3. Емельянова Н.З. Основы построения автоматизированных информационных систем : учеб. пособие: рек. Мин. обр. РФ/ Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. -М.: ФОРУМ: ИНФРА-М, 2007.-416 с.

2. Методические материалы к выполнению самостоятельной работы

1.Современные технологии объектно-ориентированного анализа и проектирования информационных систем

Концепции *объектно-ориентированного анализа и проектирования*. Эволюция и краткая характеристика основных подходов к разработке информационных моделей бизнес-систем и бизнес-процессов. Особенности проектирования, анализа и формализации корпоративных систем. Основные этапы развития языка UML и принятые стандарты. Разработчики графической нотации и специфика ее использования в процессе создания масштабируемых программных систем.

2.Основные элементы языка UML

Общая характеристика моделей *объектно-ориентированного анализа и проектирования*. Пакеты в языке UML, их графическое изображение. Базовые семантические конструкции языка, их описание с помощью специальных обозначений. Канонические диаграммы языка UML и

особенности их графического представления. Рекомендации по графическому изображению диаграмм языка UML.

3.Элементы графической нотации диаграммы вариантов использования

Диаграмма вариантов использования как концептуальное представление бизнес-системы в процессе ее разработки. Особенности графического изображения вариантов использования и актеров. Основные отношения между графическими элементами, их стереотипы. Понятия бизнесактера, сотрудника и бизнес варианта использования. Особенности концептуального моделирования бизнес-систем в форме *диаграмм вариантов использования*.

4.Спецификация требований и рекомендации по написанию эффективных вариантов использования

Классификация требований, их спецификация в форме диаграмм вариантов использования. Сценарии вариантов использования, их графическая интерпретация. Применение шаблонов сценариев при разработке диаграмм вариантов использования. Примеры написания текста сценария. Рекомендации по написанию вариантов использования.

5.Элементы графической нотации диаграммы классов

Графическое изображение класса, его атрибутов и операций. Конкретные и абстрактные классы. Видимость и кратность атрибутов и операций. Расширение языка UML для построения моделей программного обеспечения и бизнес-систем. Интерфейсы и варианты их графического обозначения.

6.Отношения и их графическое изображение на диаграмме классов

Отношение ассоциации, варианты его графического изображения. Отношение обобщения классов. Наследование атрибутов и операций классов. Отношения агрегации и композиции, их семантические особенности. Рекомендации по построению диаграмм классов.

7.Элементы графической нотации диаграммы кооперации

Назначение диаграммы кооперации. Объекты, их имена и графическое изображение. Активные и пассивные объекты, их графическое изображение. Мультиобъекты и составные объекты. Графическое изображение связей, посылаемых и принимаемых сообщений между объектами. Формат и синтаксис записи сообщений. *Стереотипы сообщений*. Рекомендации по построению диаграмм кооперации.

8.Элементы графической нотации диаграммы последовательности

Назначение диаграммы последовательности. Объекты, их графическое представление. Линия жизни и *фокус управления*. Особенности изображения моментов создания и уничтожения объектов. Ветвление и условия их выполнения. Рекомендации по построению *диаграмм последовательности*.

9.Элементы графической нотации диаграммы состояний

Особенности моделирования поведения объектов в виде диаграмм состояний. Понятие конечного автомата и логика изменения его состояний. Описание реакции объекта на асинхронные внешние события в форме диаграммы состояния. Внутренние действия состояния и ду-деятельность. Триггерные и нетриггерные переходы. События и их спецификация на диаграммах состояний.

10.Моделирование параллельного поведения с помощью диаграмм состояний

Особенности моделирования параллельного поведения объектов в форме диаграмм состояний. Понятие составного состояния и подсостояния. Сложные переходы и псевдосостояния. Исторические состояния, особенности их использования. Синхронизация параллельных подсостояний. Рекомендации по построению диаграмм состояний.

11.Элементы графической нотации диаграммы деятельности

Диаграмма деятельности и особенности ее построения. Состояния и переходы на диаграмме деятельности. Ветвление и распараллеливание процессов на диаграмме деятельности. Особенности изображения объектов на диаграмме деятельности. Использование диаграмм деятельности для описания моделей бизнес-процессов.

12.Элементы графической нотации диаграммы компонентов

Назначение *диаграммы компонентов*, ее основные элементы. Особенности физического представления программных систем. Компоненты программных систем, их разновидности. Интерфейсы, их реализация компонентами. Использование *диаграммы компонентов* для проектирования зависимостей между компонентами. Рекомендации по построению *диаграммы компонентов*.

13.Элементы графической нотации диаграммы развертывания

Диаграмма развертывания, особенности ее построения. Варианты графического изображения узлов на диаграмме развертывания. Специфика представления ресурсоемких узлов и технических устройств. Соединения и зависимости на диаграмме развертывания. Рекомендации по построению диаграммы развертывания.

14.Паттерны проектирования и их представление в нотации UML

Паттерны *объектно-ориентированного анализа и проектирования*, их классификация. *Паттерны проектирования* в нотации языка UML. Полный список *паттернов проектирования* GoF. Паттерн Фасад, его обозначение в нотации языка UML и пример реализации. Паттерн Наблюдатель, его обозначение в нотации языка UML и пример реализации

Дополнительная литература:

1 Бобровский С.И. Технологии Delphi 2006. Новые возможности/ С. И. Бобровский. -СПб.: Питер, 2006.-288 с.
2 Информационно-измерительная техника и технологии: Учеб. для вузов: Рек. Мин.обр. РФ/ В. И. Калашников, С.В.Нефедов, А.Б.Путилин; ред. Г.Г.Раннев. -М.: Высш. шк., 2002.-456 с.
3 Семенов А.С. Информационные технологии: объектно-ориентированное моделирование: Анализ и проектирование производственных систем: учеб.пособие: рек. УМО по обр./А.С.Семенов. -М.: Изд-во Моск.гос.технол. ун-та Станкин, 2001.-65 с..
4 Холмогоров В. Основы веб-мастерства: Учеб. курс/ В. Холмогоров. -2-е изд.. -СПб.: Питер, 2003.-317 с.
5 Хорошилов А.В. Мировые информационные ресурсы: Учеб.пособие: Рек. УМО по обр./ А.В. Хорошилов , С.Н.Селетков. -СПб.: Питер, 2004.-176 с.
6 Городняя Л.В. Основы функционального программирования : Курс лекций: учеб. пособие: Рек. УМО вузов/ Л.В.Городняя. -М.: Интернет- Ун-т Информ. Технологий, 2004.-273 с.
7 Кватрани Т. Визуальное моделирование с помощью Rational Rose 2002 и UML: Пер. с англ./ Т. Кватрани. -М.: Вильямс, 2003.-187 с.

Периодические издания:

1 Программирование
2 Программные продукты и системы
3 Информатика и системы управления
4 Computer – IEEE Computer Magazine

Программное обеспечение и Интернет-ресурсы:

Свободно распространяемое программное обеспечение

№ п/п	Наименование ресурса	Характеристика
1	рее http://www.iqlib.ru	Интернет библиотека образовательных изданий, в которой собраны электронные учебники, справочные и учебные пособия. Удобный поиск по ключевым словам, отдельным темам и отраслям знаний.
2	http://www.intuit.ru	Интернет-университет информационных технологий, в котором вобраны электронные и видео-курсы по отраслям знаний
3	http://amursu.ru	Сайт АмГУ, Библиотека – электронная биб-

		лиотека АмГУ
4	http://www.biblioclub.ru	Электронная библиотечная система «Университетская библиотека – online»: специализируется на учебных материалах для ВУЗов по научно-гуманитарной тематике, а так же содержит материалы по точным и естественным наукам

3. Перечень используемых программных продуктов

Windows XP; MS Office XP; BPWin; ERWin; Delphi.

4.Фонд тестовых и контрольных заданий

Оценочные средства для текущего контроля успеваемости

Вопросы к зачету:

1. Понятие жизненного цикла (ЖЦ) ПО. Международные стандарты и стандарты РФ, регламентирующие процессы разработки ПО.
2. Основные процессы ЖЦ ПО. Вспомогательные процессы ЖЦ ПО. Организационные процессы ЖЦ ПО в соответствии со стандартом ISO 12207.
3. Модели ЖЦ ПО. Подход RAD.
4. Основы структурного подхода к разработке ПО. Метод функционального моделирования. Состав функциональной модели.
5. Метод нисходящего модульного проектирования. Требования к связности модулей и к их сцеплению.
6. Метод восходящего проектирования ПС.
7. Структура SADT-модели, декомпозиция диаграмм. Типы связей между функциями. Сравнительный анализ SADT –моделей и диаграмм потоков данных (DFD).
8. Моделирование потоков данных (процессов): состав диаграмм потоков данных (DFD), построение иерархии диаграмм потоков данных. Сравнительный анализ SADT –моделей и диаграмм потоков данных (DFD).
9. Технологическая цепочка разработки ПО при соблюдении структурного подхода (модель предметной области, начальная и полная контекстные диаграммы, диаграммы «сущность-связь»).
10. Разработка контекстной и детализированных DFD в среде BPwin.
11. Спецификации потоков данных и управлений. Различные способы задания спецификаций.
12. Словарь данных. Создание словаря модели проектируемого ПО в среде BPwin.
13. Взаимообусловленность DFD и модели данных ERD. Разработка модели данных ERD с помощью пакета Computer Associates и в среде ERwin. Логический и физический уровни моделирования.
14. Сущность объектно-ориентированного подхода. Унифицированный язык моделирования UML. Общая структура языка UML (мета-метамодель, метамодель, модель, объекты пользователя). Канонические диаграммы языка UML, их классификация.
15. Диаграмма вариантов использования, её основные элементы и типы отношений между ними.
16. Диаграмма классов, имена, атрибуты, операции. Отношения между классами (отношения зависимости, ассоциации, обобщения, агрегации, композиции).
17. Диаграмма состояний: состояние (имя, список внутр. действий, нач. сост., кон. сост.); переход (событие, сторожевое условие, выражение действия), составное состояние и подсостояние, историческое состояние, сложные переходы, синхронизирующие состояния.
18. Диаграмма деятельности: состояние действия, переходы, дорожки, объекты на диаграмме деятельности.

19. Диаграмма последовательности: линия жизни объекта, фокус управления, сообщения, ветвление потока управления, стереотипы сообщений.
20. Диаграмма кооперации.
21. Диаграммы реализации: диаграмма компонентов, диаграмма развёртывания.
22. Требования, предъявляемые к сопроводительной документации ПС; состав документации.
23. Среда проектирования ПО Power Designer.