ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

АВТОМАТИЗАЦИЯ ОБРАБОТКИ БИОМЕДИЦИНСКОЙ ИНФОРМАЦИИ

Методические рекомендации к лабораторным работам для студентов направления подготовки 12.03.04 «Биотехнические системы и технологии» дневной формы обучения

Могилев 2018

УДК 004.3:60
ББК 32.973.26-018.2:53.6
А 22

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «13» марта 2018 г., протокол № 11

Составители: канд. техн. наук, доц. К. В. Захарченков;
канд. техн. наук, доц. Т. В. Мрочек

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации содержат описание десяти лабораторных работ, выполнение которых закрепляет материал, изложенный в курсе «Автоматизация обработки биомедицинской информации». Предназначены для студентов направления подготовки 12.03.04 «Биотехнические системы и технологии» дневной формы обучения.

Учебно-методическое издание

АВТОМАТИЗАЦИЯ ОБРАБОТКИ БИОМЕДИЦИНСКОЙ ИНФОРМАЦИИ

Ответственный за выпуск А. И. Якимов

Технический редактор А. Т. Червинская

Компьютерная верстка Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 16 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилев.

© ГУ ВПО «Белорусско-Российский
университет», 2018

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Содержание

Введение 4
1 Общая характеристика CASE-средства Enterprise Architect 5
2 Разработка диаграммы вариантов использования 9
3 Разработка диаграммы классов на уровне сущностей 11
4 Разработка атрибутов, операций объектов и отношений между ними на диаграмме классов 14
5 Разработка диаграммы кооперации и редактирование свойств ее элементов 16
6 Разработка диаграммы последовательности и редактирование свойств ее элементов 18
7 Разработка диаграммы состояний и редактирование свойств ее элементов 21
8 Разработка диаграммы деятельности и редактирование свойств ее элементов 23
9 Разработка диаграммы компонентов и редактирование свойств ее элементов 25
10 Разработка диаграммы развертывания и редактирование свойств ее элементов 28
Список литературы 29

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Введение

Целью дисциплины «Автоматизация обработки биомедицинской информации» является формирование профессиональных компетенций, необходимых в процессе проектирования автоматизированных систем обработки биомедицинской информации.

В современных условиях создание сложных программных приложений для автоматизации обработки больших объемов информации невозможно без использования систем автоматизированного конструирования программного обеспечения – CASE-систем (CASE – Computer Aided System/Software Engineering) [7, 8]. CASE-системы существенно сокращают сроки и затраты разработки, оказывая помощь специалисту в проведении рутинных операций, облегчая его работу на самых разных этапах жизненного цикла разработки. Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования информационных систем: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл программного обеспечения.

Для хранения и обработки информации широко используются СУБД различных типов, наибольшее распространение получили реляционные СУБД (Microsoft Office Access, Microsoft SQL Server и т. д.) [5, 7–9]. Они имеют наиболее понятный и наглядный интерфейс, вся информация хранится в виде отдельных таблиц, с которыми практически все встречались в повседневной деятельности.

В методических рекомендациях рассмотрены все этапы разработки автоматизированной системы обработки информации (АСОИ) – от определения требований к создаваемой системе до построения диаграмм ее компонентов и физического размещения на аппаратных средствах пользователя. В процессе проектирования используются средство визуального моделирования Enterprise Architect и унифицированный язык моделирования Unified Modeling Language (UML) [1, 6, 8].

В [10] изложены теоретические основы построения основных типов диаграмм UML и приведены примеры всех изучаемых типов диаграмм. В данных методических рекомендациях описаны основные приемы построения диаграмм в среде Enterprise Architect.

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

1 Общая характеристика CASE-средства Enterprise Architect

Цель: изучение основных возможностей и приемов работы в CASE-средстве Enterprise Architect.

1.1 Теоретические положения

1.1.1 Краткая характеристика Enterprise Architect.

Enterprise Architect – это мощное средство визуального моделирования, поддерживающее полный жизненный цикл создания программных систем с использованием унифицированного языка моделирования (UML).

Диаграмма в UML – это графическое представление набора элементов, изображаемое в виде связанного графа с вершинами (сущностями) и ребрами (отношениями), используемое для визуализации системы с разных точек зрения.

Enterprise Architect позволяет:

– разрабатывать диаграммы, документировать созданные модели и элементы;

– генерировать код для разрабатываемого программного обеспечения на различных языках (C#, C++, Delphi, Java, PHP, Python, Visual Basic, Visual Basic .NET и т. д.);

– импортировать коды на различных языках;

– создавать различные шаблоны моделей предметной области и систем.

В лабораторном курсе будут изучены две основные группы диаграмм UML: структурные диаграммы, использующиеся для моделирования элементов систем и их связей, и поведенческие диаграммы, использующиеся для моделирования особенностей поведения систем или бизнес-процессов предприятий.

К структурным диаграммам (structural diagrams) относятся:

– диаграмма классов (class diagram), в том числе диаграммы классов баз данных (database diagram);

– диаграмма компонентов (component diagram);

– диаграмма размещения (развертывания) (deployment diagram).

К поведенческим диаграммам (behavioral diagrams) относятся:

– диаграмма вариантов использования (use case diagram);

– диаграммы взаимодействия, среди которых выделяют диаграммы последовательности (sequence diagrams) и диаграммы кооперации (communication diagrams (или ранее – collaboration diagrams));

– диаграмма деятельности (диаграмма активностей) (activity diagram);

– диаграмма состояний (state machine diagram).

Основные элементы интерфейса Enterprise Architect. После запуска Enterprise Architect версии 12.1 на экране появляется стартовая страница, содержащая следующие возможности:

1) **Learning Center** – вызов модуля помощи, в котором содержатся полезные руководства, ссылки на веб-ресурсы компании Sparx Systems – производи-

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

теля Enterprise Architect, где содержатся уроки по языку UML и примеры;

2) **New File** – создание нового проекта *.eap;

3) **Open File** – открытие существующего файла проекта *.eap;

4) **Server Connection** – выбор имени источника данных для подключения;

5) **Cloud Connection** – выбор имени, протокола, сервера и порта для подключения к модели в облаке;

6) **Recent** – вызов моделей из текущего списка, в том числе примера модели EAExample.eap.

В правой части стартовой страницы содержатся ссылки на наиболее часто используемые ресурсы Enterprise Architect, сгруппированные по панелям Start, ViewPoint, Model, Verify, Execute, Recent, Publish, Manage, Extensions.

Браузер проекта (Project Browser) – это древовидная иерархическая структура, позволяющая осуществлять навигацию по модели. Все, что добавляется к ней, – действующие лица, варианты использования, классы, компоненты, пакеты – будет показано в окне браузера.

Браузер проекта можно разделить на представления, каждое из которых содержит диаграммы, пакеты и другие элементы.

Представление вариантов использования включает модели вариантов использования и бизнес-процессов.

Динамическое представление включает диаграммы состояний, деятельности и последовательности; логическое – модель классов.

Представление компонентов – это представление системных компонентов (исполняемые, DDL и другие компоненты).

Представление размещения – физическая модель, которая показывает, какие использованы технические средства ЭВМ и какое программное обеспечение установлено.

Главное меню включает набор команд, представленных на рисунке 1.1.

Рисунок 1.1 – Главное меню EA

Основное назначение команд главного меню:

1) **File** (файл) – создание, открытие, сохранение, закрытие проекта, настройка параметров печати, печать диаграмм;

2) **Edit** (редактирование) – отмена действия, возврат к отмененному действию, выбор элементов, копирование, вставка, поиск, удаление, пометка выбранных элементов диаграмм;

3) **View** (область просмотра) – настройка панелей инструментов, окон;

4) **Project** (проект) – поддержка управления проектом, настройки проекта (авторы, стереотипы UML, типы данных и т. п.);

5) **Package** (пакет) – создание пакета, задание свойств его отображения, вызов генератора документации по проекту (в форматах RTF, PDF, DOCX), генератор кода DDL базы данных по проекту и импорт схемы базы данных из источника и т. д.;

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

6) **Diagram** (диаграмма) – создание, сохранение изображений диаграмм, настройка свойств диаграмм;

7) **Element** (элемент) – управление элементами диаграмм;

8) **Tools** (инструменты) – управление инструментами доступа к различным настройкам для генерации кода, управление *.EAP-файлами, настройка собственной конфигурации меню и т. д.;

9) **Analyzer** (анализатор) – содержит панели отладчика и симулятора моделей, точки прерывания и маркеры и т. д.;

10) **Extensions** (расширения) – работа с внешними компонентами;

11) **Window** (окно) – работа с окнами;

12) **Help (**справка) – справка.

В меню **Tools** наиболее важным является пункт **Options**. Здесь находятся следующие опции настройки Enterprise Architect для отображения моделей и работы с элементами модели:

1) **General** – общие настройки проекта, позволяющие задать, например, автора, адрес веб-сайта, общие настройки браузера проекта;

2) **Diagram** – задание опций для новых диаграмм (размер изображения, выделяемая память и т. п.), в том числе:

– **Standard Colors** – установка цвета ряда объектов (экран, примечания, атрибуты, методы и т. п.) и их фона;

– **Diagram Behavior** – задание поведения диаграммы (автосохранение изменений, отображение линий жизни, цветная печать, автоматическая группировка элементов и т. д.);

– **Diagram Sequence** – задание настроек шрифта и фокуса контрольного индикатора для диаграммы последовательности;

3) **Objects** – задание отображения элементов на диаграммах (установка шрифта по умолчанию, толщины линий границ, тени и т. п.);

4) **Links** – настройки создания, поведения и нотации связей;

5) **Communication Colors** – установка цветов, используемых в диаграммах взаимодействия;

6) **XML Specifications** – установка настроек для работы с XML;

7) **Source Code Engineering** – задание общих настроек, применяемых для всех языков при генерации кода из Enterprise Architect.

Окно диаграммы, появляющееся после создания диаграммы требуемого типа, используется для построения диаграмм. При внесении изменений в элементы диаграммы EA автоматически обновляет браузер. Аналогично при внесении изменений в элемент с использованием браузера EA автоматически обновляет соответствующие диаграммы.

Спецификация элементов используется для документирования информации, связанной с элементами диаграмм.

Панель инструментов **Toolbox** представлена как панель с иконками, которые используются для создания элементов и связей между ними. Для каждого конкретного типа диаграмм панель имеет свое собственное представление, содержащее следующие группы: группа типовых элементов данной диаграммы, группа отношений, используемых в данной диаграмме, паттерны (шаблоны) диаграммы, группа общих элементов (примечания, текст, гиперссылки,

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

легенда диаграммы, требования и т. д.), группа артефактов (пользовательские истории, графики и т. д.).

Создание проекта в Enterprise Architect. Создание проекта является первым шагом при работе Enterprise Architect. Проект можно создавать как без использования шаблонов, так и с их использованием. Готовый проект со всеми диаграммами сохраняется в файле с расширением .eap.

Для создания проекта:

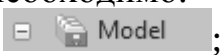
1) необходимо выбрать на стартовой станице пункт «New File»;

2) в появившемся окне следует ввести в поле «Имя файла» имя файла проекта и нажать кнопку «Сохранить»;

3) в появившемся окне Model Wizard выбора шаблона модели необходимо нажать на кнопки «None» (т. е. создается пустой проект) и «OK»;

4) на экране на стартовой странице отобразится ссылка на созданные проекты.

Создание диаграммы в Enterprise Architect. С целью упорядочивания диаграммы могут помещаться в пакеты (папки). Для добавления новой диаграммы в соответствующую область просмотра необходимо:

1) выделить в браузере проекта уровень Model 

;

2) нажать значок папки (New Package) в браузере проекта;

3) в появившемся на экране окне создания пакета задать имя пакета и определить дальнейшие действия по созданию пакета: пакет с паттерном диаграммы, пакет с созданием диаграммы или только пакет. Необходимо выбрать первый вариант действий;

4) в появившемся на экране окне Model Wizard, установив курсор в левой колонке окна на пункте Core Modeling или Database, следует выбрать требуемый тип диаграммы и нажать кнопку «OK»;

5) в браузере проекта раскрыть содержимое созданного пакета, нажав на значок , и открыть диаграмму двойным щелчком мыши по ее иконке в браузере Enterprise Architect. На экране отобразится поле для построения диаграммы и соответствующая панель инструментов **Toolbox**.

Задание

Необходимо ознакомиться с основными возможностями и освоить приемы работы в среде Enterprise Architect.

Содержание отчета: тема и цель работы; краткое описание основных возможностей Enterprise Architect.

Контрольные вопросы

1 Для чего предназначена среда Enterprise Architect?

2 Перечислить основные типы диаграмм для моделирования предметной области.

3 Перечислить основные элементы экрана.

4 Дать характеристику браузеру, окну диаграмм, панели инструментов Toolbox.

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

2 Разработка диаграммы вариантов использования

Цель: изучение технологии формирования диаграммы вариантов использования.

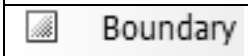
2.1 Теоретические положения

На диаграммах вариантов использования (use case diagrams) отображается взаимодействие между вариантами использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему. Этот тип диаграмм описывает общую функциональность системы. Пользователи, аналитики, разработчики могут, изучая диаграммы вариантов использования, понять, что система должна делать.

Основные элементы данной диаграммы описаны в конспекте лекций и в [1, 6, 8].

В таблице 2.1 представлены назначение и графические обозначения основных элементов диаграммы вариантов использования.

Таблица 2.1 – Основные элементы диаграммы вариантов использования

Графическое обозначение	Наименование	Назначение
	Актер (Actor)	Внешняя по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для решения определенных задач
	Вариант использования (use case)	Содержит описание, обозначающее выполнение какой-либо операции или действия
	Граница системы (boundary)	Представляет подсистему, объединяющую с целью повышения наглядности несколько вариантов использования для актера
	Пакет (package)	Сущность, используемая для семантической группировки других сущностей. Применяется для группировки диаграмм и их элементов (представляет собой аналог папки)
	Примечание (note)	Предназначено для включения в диаграмму произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемой системы (например, комментариев разработчика и ограничений)

Стандартные виды отношений между актерами и вариантами использования (ассоциации, расширения, обобщения, включения) приведены в таблице 2.2.

Чтобы создать актера или вариант использования, необходимо выделить соответствующий элемент на панели инструментов **Toolbox**, щелкнуть левой кнопкой мыши по рабочей области диаграммы, в появившемся окне ввести название элемента, при необходимости выбрать его стереотип. Новый актер

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

или вариант использования с соответствующей пиктограммой появится в браузере. Если необходимо удалить какой-либо элемент из модели, его удаляют в браузере проектов. Если удалить элемент только на диаграмме, то в браузере проектов он останется.

Таблица 2.2 – Отношения между актерами и вариантами использования

Графическое обозначение	Наименование	Назначение
	Отношение ассоциации (associate)	Устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. Может включать дополнительные условные обозначения (имя и мощность)
	Отношение расширения (extend)	Отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования. Обозначается пунктирной линией со стрелкой, помеченной ключевым словом «extend» и направленной от того варианта использования, который является расширением для исходного варианта использования
	Отношение обобщения (generalize)	Применяется, если необходимо показать, что дочерние варианты использования обладают всеми атрибутами и особенностями поведения родительских вариантов. Обозначается сплошной линией со стрелкой, которая указывает на родительский вариант использования
	Отношение включения (include)	Указывает, что поведение одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Обозначается пунктирной линией со стрелкой, помеченной ключевым словом «include» и направленной от базового варианта использования к включаемому

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Для лучшего восприятия отдельная диаграмма не должна быть перенасыщена элементами. Располагать элементы следует так, чтобы была видна логическая последовательность выполнения вариантов использования и было минимум пересечений между отношениями. Для моделирования системы можно затем разработать столько дополнительных диаграмм вариантов использования, сколько необходимо.

Добавление описаний к вариантам использования. В окне свойств варианта использования (открывается после щелчка правой клавишей мыши по изображению варианта использования и выбора из выпадающего списка пункта Properties или после двойного щелчка левой клавишей мыши по изображению варианта использования) необходимо ввести описание «Этот вариант использования дает возможность актеру выполнить следующие действия: ...».

Описание основных и альтернативных потоков событий. В окне свойств варианта использования необходимо щелкнуть левой клавишей мыши по пиктограмме (Linked Document), создать и сохранить текстовый файл в

формате RTF, прикрепленный к данному варианту использования. Текстовый файл должен содержать описание основных и альтернативных потоков событий, предусловий и постусловий так, как показано в примере диаграммы вариантов использования в [10].

Задание

Необходимо разработать в Enterprise Architect диаграмму вариантов использования по индивидуальному заданию на основе описания предметной области, полученного в практических работах № 1 и 2 [10] с учетом [2–4].

Содержание отчета: тема и цель работы; диаграмма вариантов использования, разработанная в Enterprise Architect, с описанием основных и альтернативных потоков событий, предусловий и постусловий.

Контрольные вопросы

1 Каково назначение диаграммы вариантов использования?

2 Что такое поток событий, вариант использования, актер?

3 Перечислить виды связей между вариантами использования и действующими лицами.

4 Описать технологию формирования диаграммы вариантов использования.

3 Разработка диаграммы классов на уровне сущностей

Цель: изучение основных понятий и элементов диаграммы классов.

3.1 Теоретические положения

Диаграммой классов (class diagram) называется диаграмма, на которой показан набор классов, а также отношений между ними. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. На диаграммах классов могут указываться ограничения целостности, которые должны поддерживаться в проектируемой БД.

Основные элементы данной диаграммы описаны в таблице 3.1 и более подробно – в конспекте лекций и в [1, 6, 8]. На данной диаграмме необходимо разместить классы, предварительно выделенные и описанные в практической работе № 3 [10].

Класс является абстрактным описанием или представлением свойств множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который разделен горизонтальными линиями на секции.

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

В этих секциях указываются имя класса, атрибуты (переменные) и операции (методы).

Таблица 3.1 – Основные элементы диаграммы классов

Графическое обозначение	Наименование	Назначение
	Пакет (package)	Сущность, используемая для семантической группировки других сущностей
	Класс (class)	Шаблон, на основе которого создаются однотипные объекты (файлы, материальные объекты)
	Ассоциация (association)	Отношение ассоциации соответствует наличию произвольного отношения между классами
	Зависимость (dependency)	Изображается пунктирной линией со стрелкой, направленной от клиента (зависимого элемента) к источнику (независимому элементу), и служит для представления только факта наличия связи, когда изменение одного элемента модели оказывает влияние или приводит к изменению другого элемента модели
0 *	Мощность связи (multiplicity)	Мощность (кратность) связи – это количество объектов, принимающих участие в связи. Мощность указывается на каждом конце связи. Кратность можно задать равной единице (1), указать диапазон: «ноль или единица» (0..1), «много» (0..*), «единица или больше» (1..*). Разрешается указывать определенное число (например, 5). Символ «*» обозначает произвольное конечное целое неотрицательное число, значение которого неизвестно на момент задания соответствующего отношения ассоциации

Имя класса должно быть уникальным в пределах пакета, который может содержать несколько диаграмм классов. Имя класса записывается по центру самой верхней секции полужирным шрифтом и должно начинаться с заглавной буквы. В качестве имен классов используются существительные, записанные без пробелов.

Атрибут класса служит для представления отдельного свойства или признака, который является общим для всех объектов данного класса. Атрибуты класса записываются во второй секции прямоугольника класса.

Общий формат записи отдельного атрибута класса следующий:

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

Квантор видимости (visibility) может принимать одно из четырех возможных значений и, соответственно, отображается при помощи специальных символов.

Символ «+» обозначает атрибут с областью видимости типа общедоступный (public). Атрибут с этой областью видимости доступен или виден из любо-

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

го другого класса пакета, в котором определена диаграмма.

Символ «#» обозначает атрибут с областью видимости типа защищенный (protected). Атрибут с этой областью видимости недоступен или невиден для всех классов, за исключением подклассов данного класса.

Символ «—» обозначает атрибут с областью видимости типа закрытый (private). Атрибут с этой областью видимости недоступен или невиден для всех классов без исключения.

И, наконец, символ «~» обозначает атрибут с областью видимости типа пакетный (package). Атрибут с этой областью видимости недоступен или невиден для всех классов за пределами пакета, в котором определен класс-владелец данного атрибута.

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

Имя атрибута представляет собой строку текста, которая используется в качестве идентификатора соответствующего атрибута и поэтому должна быть уникальной в пределах данного класса. Имя атрибута является единственным обязательным элементом синтаксического обозначения атрибута, оно должно начинаться со срочной (малой) буквы и не должно содержать пробелов. Тип атрибута определяется типом данных, определенным разработчиком в зависимости от языка программирования, который предполагается использовать для реализации данной модели. Типу атрибута должно предшествовать двоеточие.

Исходное значение служит для задания некоторого начального значения для соответствующего атрибута в момент создания отдельного экземпляра класса.

Задание

Необходимо разместить в Enterprise Architect сущности диаграммы классов уровня пакетов, диаграммы классов пакета «Формы», диаграммы классов пакета «Отчеты», диаграммы классов пакета «Запросы».

Содержание отчета: тема и цель работы; краткое описание основных элементов диаграммы классов; краткое описание основных сущностей.

Контрольные вопросы

1 Каково назначение диаграммы классов? Каково назначение диаграммы классов базы данных?

2 Перечислить стереотипы классов.

3 Перечислить и охарактеризовать виды связей между классами.

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

4 Разработка атрибутов, операций объектов и отношений между ними на диаграмме классов

Цель: изучение приемов создания атрибутов, операций объектов и отношений между ними на диаграмме классов в Enterprise Architect.

4.1 Теоретические положения

Основные элементы диаграммы классов (в частности, диаграммы классов базы данных (database diagram)) описаны в конспекте лекций и в [1, 6, 8].

Рассмотрим этапы создания диаграммы классов базы данных.

1 Сразу после сохранения файла проекта откроется диалоговое окно «Model Wizard» для выбора типа диаграмм.

2 В данном окне выбирается вкладка диаграмм баз данных «Database».

3 Поскольку типы данных полей таблиц базы данных зависят от той системы управления базами данных (СУБД), которая используется для ее создания, то удобно перед работой указать тип используемой для данной базы данных СУБД, например,

, и нажать кнопку «OK». Enterprise Architect поддерживает импорт и экспорт схем баз данных в MS Access, MS SQL Server, MySQL, Oracle, Firebird/InterBase, Informix и т. д.

4 В браузере проектов появится файл модели данных для MS Access, который можно открыть двойным нажатием левой кнопкой мыши по соответствующей надписи.

5 Если выбрать пункт меню Project → Settings → Database Datatypes, то в открывшемся диалоговом окне можно увидеть перечень типов данных для выбранной СУБД и сделать данные настройки настройками по умолчанию (Set as default).

6 Спроектированные на практическом занятии по соответствующей теме перечни таблиц, форм и запросов базы данных удобно разместить в соответствующие пакеты на диаграмме классов базы данных (см. пример в [10]).

7 В пакете для таблиц следует перейти к соответствующему файлу, разместить в окне таблицу. Для таблицы необходимо задать наименование. Если щелкнуть правой клавишей мыши по изображению таблицы и выбрать из выпадающего списка пункт Columns, появится окно Table Columns and Constraints, в котором можно указать все колонки таблицы и их типы (в соответствии с типами выбранной СУБД), первичные ключи и разрешить при необходимости значения NULL. Для указания значений, вставляемых в поля по умолчанию, можно использовать поле «Initial: ».

8 После окончания проектирования в таблице:

– в группе «column» будут показаны все поля таблицы;
– первичный ключ будет помечен атрибутом «PK»;
– обязательные для заполнения поля будут помечены символом «*»;
– группа «PK» будет содержать общее описание первичного ключа.

9 Аналогичным образом следует разместить в окне все остальные таблицы

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

и заполнить их.

10 Для установки связей между таблицами выбирается тип связи «Association» (связь ассоциации), которая является самым общим типом связи (в дальнейшем ее можно детализировать). Следует щелкнуть правой клавишей мыши по одной таблице и, не отпуская ее, провести линию к другой таблице.

11 Для определения атрибутов, с помощью которых связаны таблицы, следует щелкнуть правой кнопкой мыши по связи и в контекстном меню выбрать пункт «Foreign keys...». Откроется диалоговое окно «Foreign Key Constraint». Следует указать первичный ключ одной таблицы и первичный ключ второй таблицы, который станет одновременно и внешним ключом данной таблицы. Далее выбирается тип связи, например, «1 – 0..*», т. е. одной записи первой таблицы может соответствовать ноль и более записей второй таблицы. Для установления ограничений для связи (каскадное обновление или удаление) можно воспользоваться соответствующими переключателями. Далее следует сохранить настройки связи.

Рассмотрим возможность генерации SQL-скрипта для создания базы данных по модели. Необходимо выбрать пункт меню «Package» → «Database Engineering → Generate Package DDL…». Откроется диалоговое окно «Generate DDL», в котором можно установить ряд параметров. Корневым пакетом является вся диаграмма «Database MSAccess 2007» (это определяется автоматически). Опции генерации кода можно просмотреть на вкладке «Options». Выбирается способ записи в один файл «Single File» и указывается к нему путь. После этого следует нажать кнопку «Generate» для автоматической генерации SQL-скрипта. Появится диалоговое окно, отображающее ход генерации файла. Данный файл можно использовать в СУБД SQL Server для автоматического создания базы данных.

Также Enterprise Architect позволяет создавать достаточно подробную документацию по разрабатываемому приложению. Создадим, например, документ, описывающий модель данных (диаграмма базы данных). Для этого перейдем на диаграмму «Data Model» и выберем пункт меню «Package → Documentation → Document Generator (RTF/PDF/DOCX)». Откроется диалоговое окно «Generate Documentation», в котором задаются параметры документа. В поле «Output Format» указывается необходимый формат отчета – RTF, PDF или DOCX. В поле «Output to File» указывается путь к выходному файлу. Далее следует нажать кнопку «Generate» для генерации документа. После того, как документ будет создан, он появится в рабочем окне в среде Enterprise Architect.

Задание

Необходимо разработать в Enterprise Architect по индивидуальному заданию требуемые атрибуты, операции и отношения диаграммы классов уровня пакетов, диаграммы классов пакета «Формы», диаграммы классов пакета «Отчеты», диаграммы классов пакета «Запросы». Таблицы модели данных АСОИ должны находиться в третьей нормальной форме.

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Содержание отчета: тема и цель работы; краткое описание основных элементов диаграммы; диаграммы классов и их описание.

Контрольные вопросы

1 Каково назначение диаграммы классов и диаграммы классов базы данных?
2 Охарактеризовать стереотипы классов.
3 Охарактеризовать связи между классами, привести примеры.

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

5 Разработка диаграммы кооперации и редактирование свойств ее элементов

Цель: изучение основ создания в Enterprise Architect диаграмм кооперации.

5.1 Теоретические положения

На диаграмме кооперации (communication diagram) изображаются только отношения между объектами, играющими определенные роли во взаимодействии, а последовательность взаимодействий и параллельных потоков определяется с помощью порядковых номеров. Главная особенность данной диаграммы заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии. Кооперация часто реализует некоторый паттерн (шаблон проектирования).

Помимо общих элементов (экземпляров актеров, объектов и сообщений), между участниками взаимодействия отображаются ненаправленные ассоциации, над которыми указываются передаваемые ими сообщения. Другой особенностью является использование в спецификации сообщений нумерации, отражающей порядок их выполнения.

Основные элементы данной диаграммы представлены в таблице 5.1 и более подробно описаны в конспекте лекций и в [1, 6, 8].

Далее приводятся возможные варианты записи строки текста в прямоугольнике объекта.

– : C – анонимный объект, образуемый на основе класса C;
– / R – анонимный объект, играющий роль R;
– / R : C – анонимный объект, образуемый на основе класса C и играющий роль R;
– O / R – объект с именем O, играющий роль R;
– O : C – объект с именем O, образуемый на основе класса C;
– O / R : C – объект с именем O, образуемый на основе класса C и играющий роль R;
– O – объект с именем O;
– O : – «объект-сирота» с именем O;

– / R – роль с именем R;
– : C – анонимная роль на базе класса C;
– / R : C – роль с именем R на основе класса C.

Таблица 5.1 – Основные элементы диаграммы кооперации

Графическое обозначение	Наименование	Назначение
	Актер (actor)	Внешняя по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для решения определенных задач
	Объект (object)	Является отдельным экземпляром класса, который создается на этапе выполнения программы. Он может иметь свое собственное имя и конкретные значения атрибутов
	Связь (association)	Изображается отрезком прямой линии, соединяющей два и более объектов. На каждом из концов этой линии могут быть явно указаны имена ролей данной ассоциации. Рядом с линией в ее средней части может записываться имя соответствующей ассоциации. Связи не имеют собственных имен, поскольку полностью идентичны как экземпляры ассоциации. Для связей не указывается также и кратность
1: текст сообщения	Сообщение	Сообщения определяют коммуникацию между двумя объектами, один из которых передает другому некоторую информацию. При этом первый объект ожидает, что после получения сообщения вторым объектом последует выполнение некоторого действия. Т. о., именно сообщение является причиной или стимулом для начала выполнения операций, отправки сигналов, создания и уничтожения отдельных объектов. Связь обеспечивает канал для направленной передачи сообщений между объектами от объекта-источника к объекту-получателю

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Связь обеспечивает канал для направленной передачи сообщений между объектами от объекта-источника к объекту-получателю.

Связь может иметь некоторые стереотипы, которые записываются рядом с одним из ее концов и указывают на особенность реализации данной связи. В языке UML для этой цели могут использоваться следующие стереотипы:

– «association» – ассоциация (предполагается по умолчанию, поэтому данный стереотип можно не указывать);

– «parameter» – параметр метода. Соответствующий объект может быть только параметром некоторого метода;

– «local» – локальная переменная метода. Ее область видимости ограничена только соседним объектом;

– «global» – глобальная переменная. Ее область видимости распространяется на всю диаграмму кооперации;

– «self» – рефлексивная связь объекта с самим собой, которая допускает

передачу объектом сообщения самому себе. На диаграмме кооперации рефлексивная связь изображается петлей в верхней части прямоугольника объекта.

Задание

Разработать диаграмму кооперации по индивидуальному заданию.

Содержание отчета: тема и цель работы; краткое описание основных элементов диаграммы; кооперативная диаграмма в Enterprise Architect для вариантов использования по заданию.

Контрольные вопросы

1 Каково назначение кооперативной диаграммы?
2 Что такое анонимный класс?
3 Что такое пассивный и активный объекты?
4 Какие особенности имеют сообщения в диаграммах кооперации?
5 Каковы основные рекомендации по построению диаграмм кооперации?

6 Разработка диаграммы последовательности и редактирование свойств ее элементов

Цель: изучение основ создания диаграмм последовательности в Enterprise Architect.

6.1 Теоретические положения

Диаграмма последовательности (sequence diagram) показывает поведение набора взаимодействующих объектов и отражает поток событий, происходящих в рамках варианта использования. Каждый объект имеет направленную вниз линию жизни, которая отражает порядок событий для этого объекта. Сообщения посылаются от одной линии жизни к другой.

Основные элементы данной диаграммы представлены в таблице 6.1 и более подробно описаны в конспекте лекций и в [1, 6, 8].

Сообщения могут быть следующих видов (виды задаются в свойствах сообщения с помощью стереотипов):

→ – ***синхронное*** сообщение (synchronous message). Клиент посылает сообщение серверу и ждет, пока тот примет и обработает сообщение. Как правило, один объект передает синхронное сообщение второму, второй – третьему и т. д., образуя вложенный поток сообщений. В любом случае клиент, инициирующий поток сообщений, должен дождаться его завершения, т. е. возврата управления. Это самый распространенный тип сообщений;

→ – ***асинхронное*** сообщение (asynchronous message). Клиент посылает сообщение серверу и, не дожидаясь ответа, продолжает выполнять

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

следующие операции;

– ***возвращающее*** сообщение (reply message), обозначающее возврат значения или управления от сервера обратно клиенту. Стрелки этого вида зачастую отсутствуют на диаграммах, поскольку неявно предполагается их существование после окончания процесса выполнения операции.

Таблица 6.1 – Основные элементы диаграммы последовательности

Графическое обозначение	Наименование	Назначение
Lifeline	Линия жизни (lifeline)	Служит для обозначения периода времени, в течение которого объект существует в системе и, следовательно, может потенциально участвовать во всех ее взаимодействиях
	Символ уничтожения (delete lifecycle)	Используется для обозначения момента уничтожения объекта в языке UML. Ниже этого символа пунктирная линия не изображается, поскольку соответствующего объекта в системе уже нет, и этот объект должен быть исключен из всех последующих взаимодействий. Задается в окне «Message properties» в секции «Control Flow Type»
Fragment	Фрагмент (fragment)	Используются для моделирования особенностей взаимодействия (условных операторов, циклов и т. п.) вместо вложенных фокусов управления
	Сообщение (message)	Позволяет моделировать взаимодействие между объектами
	Рефлексивное сообщение (self-message)	Отображает факт посылки объектом сообщения самому себе (вызывать собственные методы)

Также часто встречаются два вида сообщений – на создание и уничтожение объектов. Первое отображается как возвращающее сообщение со стереотипом «create», второе – как синхронное сообщение со стереотипом «destroy». После получения сообщения на уничтожение объекта его линия жизни заканчивается символом .

Каждое сообщение должно иметь имя по одному из следующих вариантов:

– произвольная строка текста. Применяется на начальных стадиях проектирования или концептуальных диаграммах;

– «create» – возвращающее сообщение, требующее создания объекта;

– «destroy» – синхронное сообщение с требованием уничтожить соответствующий объект;

– «call» – синхронное сообщение, требующее выполнения операции принимающего объекта;

– «send» – асинхронное сообщение, обозначающее посылку сигнала серверу;

– «return» или «reply» – возвращающее сообщение;

– указание спецификации вызываемого метода объекта-получателя в формате

[переменная =] имя([список параметров]) [:возвращаемое значение].

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

UML определяет следующие типы фрагментов:

– alt (alternatives) – вызовы альтернативных сообщений (выполнение взаимоисключающих операций). Альтернативные сообщения (группы сообщений) отделяются друг от друга горизонтальными штриховыми линиями. Используется для моделирования условного оператора (if-then-else) и операторов выбора (case или switch);

– opt (option) – вызов дополнительного сообщения (группы сообщений) при некотором условии. Аналогичен фрагменту с типом «alt» для случая, когда используется сокращенный условный оператор (if-then);

– par (parallel) – параллельная обработка сообщений. Параллельно обрабатываемые сообщения (группы сообщений) отделяются друг от друга горизонтальными штриховыми линиями;

– loop – циклическая обработка сообщений. Используется для моделирования циклов;

– break – досрочное прерывание обработки сообщений при некотором условии. Используется как составная часть других фрагментов (как правило, «loop»);

– critical – эксклюзивно обрабатываемое сообщение (группа сообщений). Используется как составная часть других фрагментов (как правило, «par»). Подразумевает приостановку обработки любых сообщений в более общем фрагменте на время обработки сообщений внутри подфрагмента «critical»;

– neg (negative) – сообщение или событие, сгенерированное в результате невозможности обработки другого принятого сообщения. Например, если при запросе пароля getPassword() истекло время на его ввод, то вместо возврата пароля будет сгенерировано сообщение «время вышло» (англ. «timeout»);

– assert (assertion) – сообщение (группа сообщений), выполняемое после предварительной проверки некоторого условия. Если условие отрицательно, то сообщение не посылается. В программировании такой прием часто используется для локализации ошибок;

– strict – строгая последовательная обработка сообщений. Последовательно обрабатываемые сообщения (группы сообщений) отделяются друг от друга горизонтальными штриховыми линиями и обрабатываются строго по очереди сверху вниз;

– seq (sequencing) – нестрогая последовательная обработка сообщений. Сообщения (группы сообщений) отделяются друг от друга горизонтальными штриховыми линиями и могут обрабатываться в произвольном порядке, за исключением сообщений, принимаемых одним объектом;

– ignore – игнорирование сообщений. После слова «ignore» в фигурных скобках перечисляются сообщения, возникновение которых во фрагменте потенциально возможно наряду с явно отображенными и которые должны быть проигнорированы;

– consider – игнорирование других сообщений. После слова «consider» в фигурных скобках перечисляются сообщения, которые явно отображены во фрагменте, а также возникновение которых во фрагменте потенциально воз-

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

можно наряду с явно отображенными. Остальные потенциально возможные сообщения должны быть проигнорированы.

Задание

Спроектировать в Enterprise Architect диаграммы последовательности для вариантов использования по индивидуальному заданию.

Содержание отчета: тема и цель работы; краткое описание основных элементов диаграммы; диаграммы последовательности в Enterprise Architect для двух вариантов использования.

Контрольные вопросы

1 Назначение диаграммы последовательности.

2 Что такое измерения, линии жизни объектов, фокус управления?

3 Перечислить и охарактеризовать основные виды сообщений.

4 Каковы основные рекомендации по построению диаграмм последовательности?

7 Разработка диаграммы состояний и редактирование свойств ее элементов

Цель: изучение основ создания диаграммы состояний.

7.1 Теоретические положения

Диаграмма состояний (state machine diagram) используется для описания поведения, реализуемого в рамках варианта использования, или поведения экземпляра сущности (класса, объекта, компонента, узла или системы в целом). Целью является описание возможных последовательностей состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

Основные элементы данной диаграммы представлены в таблице 7.1 и более подробно описаны в конспекте лекций и в [1, 6, 8]. Пример диаграммы приведен в [10].

При построении диаграммы необходимо учитывать следующие рекомендации:

1) объект в каждый момент может находиться только в единственном состоянии. Если это не так, то это может быть как следствием ошибки, так и неявным признаком наличия параллельности у поведения моделируемого объекта. В последнем случае следует явно специфицировать необходимое число подавтоматов, вложив их в то составное состояние, которое характеризуется нарушением условия одновременности;

2) для облегчения восприятия диаграммы рекомендуется использовать де-

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

композицию со скрытием составных состояний;

3) два перехода из одного состояния не могут сработать одновременно (требование отсутствия конфликтов у переходов);

4) диаграмма не должна содержать изолированных состояний и переходов.

Таблица 7.1 – Основные элементы диаграммы состояний

Графическое обозначение	Наименование	Назначение
1	2	3
	Состояние (state)	Отображает ситуацию в ходе жизни экземпляра сущности, когда эта ситуация удовлетворяет некоторому условию, экземпляр выполняет некоторые операции или ждет наступления некоторого события
	Начальное (initial)	Начальное состояние автомата, начальное подсостояние составного состояния или параллельного подавтомата. Из начального состояния могут только исходить переходы
	Конечное (final)	Конечное состояние автомата, конечное подсостояние составного состояния или параллельного подавтомата. В конечное состояние могут только входить переходы
	Немедленное завершение (terminate)	Аналогично конечному состоянию, но подразумевает немедленное прекращение деятельности и уничтожение экземпляра сущности, для которой построен автомат
	Точка входа (entry point)	Точка входа в автомат или составное состояние. Может быть несколько. Допускается крепление к границе составного состояния
	Точка выхода (exit point)	Точка выхода из автомата или составного состояния. Может быть несколько. Допускается крепление к границе составного состояния
Fork/Join Fork/Join	Ветвление / Соединение (fork / join)	Ветвление переходов в параллельных подавтоматах
	Переход (junction)	Соединение и ветвление переходов для последовательных состояний. В случае ветвления для каждой исходящей из перехода ассоциации должно быть задано сторожевое условие, например: [A > B] [A <= B] [A > B] [A <= B]
	Выбор (choice)	Аналогично переходу, работающему на ветвление для последовательных состояний: ID [> 100] [<= 100] [ID > 100] [ID <= 100]
	Символ состояния со скрытыми подсостояниями (state mashine)	Используется для обозначения составного состояния, имеющего внутри себя скрытые (не показанные на диаграмме) подсостояния

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Окончание таблицы 7.1

1	2	3
	Переход (transiton)	Отношение между двумя состояниями, показывающее возможный путь изменения состояния экземпляра сущности. Считается, что в состоянии экземпляр сущности находится продолжительное время, а переход выполняется мгновенно

Задание

Необходимо разработать в Enterprise Architect диаграмму состояний АСОИ по индивидуальному заданию.

Содержание отчета: тема и цель работы; краткое описание основных элементов диаграммы; диаграмма состояний АСОИ.

Контрольные вопросы

1 Каково назначение диаграммы состояний?
2 Какие виды операций различают в UML?
3 Что такое состояние, переход, событие, автомат, историческое состояние, характеристика состояния, метка состояния?
4 Охарактеризовать виды переходов.
5 Перечислить рекомендации по построению диаграммы состояний.

8 Разработка диаграммы деятельности и редактирование свойств ее элементов

Цель: изучение основ создания диаграммы деятельности в Enterprise Architect.

8.1 Теоретические положения

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности (activity diagrams). Основные элементы данной диаграммы представлены в таблице 8.1 и более подробно описаны в конспекте лекций и в [1, 6, 8].

Таблица 8.1 – Основные элементы диаграммы деятельности

Графическое обозначение	Наименование	Назначение
1	2	3
	Деятельность (activity)	Составная (неатомарная) операция, реализуемая экземпляром в конкретном состоянии, выполнение которой может быть прервано

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Окончание таблицы 8.1

1	2	3
	Действие (action)	Атомарная операция, выполнение которой не может быть прервано, приводящая к смене состояния или возвращающая значение
	«Макро-» деятельность (structured activity)	Используется для отображения обобщенной деятельности, которая декомпозируется. Фактически в деятельность вписывается имя другой диаграммы, имеющей определенную внутреннюю структуру
	Объект (object)	Объект системы или предметной области (экземпляр класса). Используется для отображения ролевой ответственности при выполнении какой-либо деятельности, для отображения объектов реального мира, связанных с бизнес-процессом. Объекты могут быть использованы с различными стереотипами, например, «worker», «entity»
	Отправка сигнала (send signal)	Если в результате действия генерируется сигнал для последующей обработки (из символа действия исходит стрелка и входит в символ сигнала), то он отображается как «отправка сигнала»
	Прием сигнала (accept signal)	Когда сигнал поступает на обработку (из символа сигнала исходит стрелка и входит в символ действия), то он отображается как «прием сигнала»
	Поток управления (control flow)	Представляет собой самый общий вид перехода и задает порядок выполнения операций. Используется для отображений связей между деятельностями
	Объектный поток (object flow)	Применяется, когда на диаграмме необходимо, помимо передачи управления, отобразить и передачу информации. Используется для отображения связей между деятельностью и объектом
	Поток прерывания (interruptible flow)	Исходит из символа «прием сигнала», расположенного в прерываемой области, и входит в действие – обработчик прерывания
	Начальный узел (initial node)	Используется для отображения начала бизнес-процесса
	Завершение деятельности (activity final)	Используется для отображения окончания бизнес-процесса
	Завершение потока (flow final)	Точка завершения потока
	Решение (Decision)	Позволяет отобразить разветвление или слияние потока работ или вычислительного процесса. Исходящие из него стрелки помечаются сторожевыми условиями ветвления
Fork/Join Fork/Join	Ветвление / Соединение (fork / join)	Используется для слияния и расщепления параллельных потоков

При построении диаграммы принимают во внимание следующие рекомендации:

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

1) учет классических принципов моделирования – декомпозиции и иерархического упорядочения. То есть при моделировании алгоритма вначале строится контекстная диаграмма с деятельностями, которые после детализируются с помощью соответствующих диаграмм декомпозиции;

2) минимум числа пересечений линий;

3) если на диаграмме имеется ветвление на параллельные или альтернативные потоки, то должно указываться и соответствующее соединение этих потоков. При использовании альтернативных потоков каждый из них должен быть специфицирован с помощью сторожевого условия. Сторожевые условия не должны допускать одновременного срабатывания двух и более переходов.

Задание

Необходимо разработать в Enterprise Architect диаграмму деятельности АСОИ по варианту индивидуального задания.

Содержание отчета: тема и цель работы; краткое описание основных элементов диаграммы; диаграмма деятельности АСОИ.

Контрольные вопросы

1 Каково назначение диаграммы деятельности?

2 Каков состав диаграммы деятельности?

3 Что такое состояние действия, переходы, сторожевое условие, дорожки, исполняемые узлы, управляющие узлы, коннекторы?

9 Разработка диаграммы компонентов и редактирование свойств ее элементов

Цель: изучение основных понятий, используемых при создании диаграммы компонентов.

9.1 Теоретические положения

Диаграмма компонентов (component diagram) позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Диаграмма обеспечивает согласованный переход от логического к физическому представлению системы в виде программных компонентов. Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения.

Основные элементы данной диаграммы представлены в таблице 9.1 и более подробно описаны в конспекте лекций и в [1, 6, 8].

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Таблица 9.1 – Основные элементы диаграммы компонентов

Графическое обозначение	Наименование	Назначение
	Пакет (package)	Сущность, используемая для семантической группировки других сущностей
	Компонент (component)	Компонент реализует некоторый набор интерфейсов и служит для общего обозначения элементов физического представления модели. Компоненты представляют собой файлы с исходным кодом классов, библиотеки, исполняемые модули и т. п., которые должны обладать согласованным набором интерфейсов. Внутри компонента, как и класса, могут быть выделены дополнительные секции, в которых указываются предоставляемые (provided) или необходимые для работы (required) интерфейсы и классы, методы (operations), наименование файла-компонента (artifacts) и т. п.
	Класс (class)	Шаблон, на основе которого генерируются (создаются) однотипные объекты
	Интерфейс (interface)	Внешне видимый, именованный набор операций, который класс, компонент или подсистема может предоставить другому классу, компоненту или подсистеме для выполнения им своих функций. В некоторых языках программирования (например, Java) интерфейс представляет собой отдельный класс, включаемый и реализуемый (конкретизируемый) в части программного кода операций в составе других классов. На диаграмме компонентов интерфейс отображается так же, как и на диаграмме классов (слева от компонента – необходимые для работы интерфейсы, справа – предоставляемые)
	Объект (object)	Объект системы (экземпляр класса)
	Отношение ассоциации (associate)	Отображается между компонентами и их интерфейсами
	Отношение зависимости (dependency)	Изображается пунктирной линией со стрелкой, направленной от клиента (зависимого элемента) к источнику (независимому элементу), и служит для представления только факта наличия связи, когда изменение одного элемента модели оказывает влияние или приводит к изменению другого элемента модели

В языке UML выделяют три вида компонентов:

– компоненты развертывания, которые обеспечивают непосредственное выполнение системой своих функций: динамически подключаемые библиотеки с расширением dll, Web-страницы на языке разметки гипертекста с расширением html и файлы справки с расширением hlp;

– компоненты-рабочие продукты: файлы с исходными текстами программ, например, с расширениями h или cpp для языка C++;

– компоненты исполнения, представляющие исполнимые модули – файлы с расширением exe.

Компоненты могут иметь следующие стандартные стереотипы:

– «file» – любой файл, кроме таблицы;

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

– «executable» – программа (исполняемый файл);
– «library» – статическая или динамическая библиотека;
– «source» – файл с исходным текстом программы;
– «document» – остальные файлы (например, файл справки);
– «table» – таблица базы данных.

Отношение зависимости означает зависимость реализации одних компонентов от реализации других. Такое возможно в следующих случаях:

– в методах классов одного компонента (зависимого) осуществляется вызов методов или обращение к атрибутам классов другого компонента (независимого);

– компонент состоит из других компонентов (например, при сборке исполняемого файла из файлов с исходными кодами);

– компонент осуществляет чтение или запись данных в другой компонент;

– связь между таблицами БД.

При построении диаграммы принимают во внимание следующие рекомендации:

1) до начала разработки необходимо принять решения о выборе операционной системы, программных сред, языков программирования и базы данных, на которых предполагается реализовывать систему;

2) необходимо решить, из каких физических частей (файлов) будет состоять программная система;

3) рекомендуется минимизировать количество связей между исполняемыми компонентами, а необходимое взаимодействие осуществлять через хранимую в единой базе данных информацию.

Задание

Необходимо разработать в Enterprise Architect диаграмму компонентов АСОИ по варианту индивидуального задания.

Содержание отчета: тема и цель работы; краткое описание основных элементов диаграммы; диаграмма компонентов АСОИ.

Контрольные вопросы

1 Каково назначение диаграммы компонентов?

2 Перечислить и охарактеризовать типы компонентов. Назвать стандартные стереотипы компонентов.

3 Какие отношения могут быть представлены на диаграмме компонентов?

4 Перечислить основные рекомендации по построению диаграммы компонентов.

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

10 Разработка диаграммы развертывания и редактирование свойств ее элементов

Цель: изучение основ создания диаграммы развертывания.

10.1 Теоретические положения

Если диаграммы компонентов показывают, как выглядит модель на физическом уровне, то диаграмма развертывания (размещения) (deployment diagram) отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она применяется для представления общей конфигурации и топологии распределенной информационной системы, содержит сведения о распределении компонентов по отдельным узлам системы и каналом связи между аппаратными средствами (соединения).

Диаграмма развертывания предназначена для визуализации элементов и компонентов системы, существующих лишь на этапе ее исполнения (runtime), к которым относятся исполняемые файлы, динамические библиотеки, таблицы базы данных и т. д. Компоненты, не используемые на этапе исполнения (например, исходные тексты программ), не показываются.

Основные цели, преследуемые при разработке данной диаграммы:

– распределение компонентов системы по ее физическим узлам;

– отображение физических связей между узлами системы на этапе исполнения;

– выявление узких мест системы и реконфигурация ее топологии для достижения требуемой производительности.

Основные элементы данной диаграммы представлены в таблице 10.1 и более подробно описаны в конспекте лекций и в [1, 6, 8].

Таблица 10.1 – Основные элементы диаграммы развертывания

Графическое обозначение	Наименование	Назначение
	Узел (node)	Представляет собой некоторый физически существующий элемент системы. В качестве узла могут рассматриваться компьютеры, датчики, принтеры, модемы, цифровые камеры, сканеры и т. д. Графически узел изображается в форме трехмерного куба, внутри которого указываются его имя и, возможно, дополнительная информация в виде помеченного значения
	Прибор (device)	Используется для отображения аппаратных средств и механических устройств
	Соединение (connection)	Соединения показываются в виде ассоциации и изображаются линиями без стрелок. Наличие такой линии указывает на необходимость организации канала для обмена информацией между соответствующими узлами, например, оптоволоконная линия или спутниковая связь, ссылка на сайт в сети Интернет и т. д.
	Отношение зависимости (dependency)	Изображается между узлом и развернутыми на нем компонентами. Используется при большом количестве развернутых на узле компонентов

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Рекомендации по построению диаграммы:

– перед разработкой диаграммы необходимо идентифицировать:

а) категории пользователей, для каждой из которых должны быть определены количество пользователей и требуемые компоненты системы;

б) аппаратные, технические и другие типы устройств, необходимые для выполнения системой своих функций;

в) виды и требуемую пропускную способность каналов связи;

– для наглядного представления распределенной информационной системы рекомендуется отображать компоненты, интерфейсы и связи между ними;

– для наглядного отображения специфики узлов вместо стандартного символа узла со строковым стереотипом могут использоваться графические стереотипы;

– диаграмму компонентов и размещения можно объединить и представить на одном рисунке.

Задание

Необходимо разработать в Enterprise Architect диаграмму развертывания по варианту индивидуального задания.

Содержание отчета: тема и цель работы; краткое описание основных элементов диаграммы; диаграмма развертывания.

Контрольные вопросы

1 Каково назначение диаграммы развертывания?

2 Как изображаются основные компоненты диаграммы?

3 Перечислить основные рекомендации по разработке диаграмм компонентов и развертывания.

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/

Список литературы

1 **Буч, Г.** Введение в UML от создателей языка / Г. Буч, Д. Рамбо, И. Якобсон ; пер. с англ. Н. Мухина. – 2-е изд. – Москва : ДМК Пресс, 2012. – 494 с. : ил.

2 **Ершов, Ю. А.** Биотехнические системы медицинского назначения : учебник для бакалавриата и магистратуры : в 2 ч. Ч. 1 : Количественное описание биообъектов / Ю. А. Ершов, С. И. Щукин. – 2-е изд., испр. и доп. – Москва : Юрайт, 2017. – 180 с.

3 **Ершов, Ю. А.** Биотехнические системы медицинского назначения : учебник для бакалавриата и магистратуры : в 2 ч. Ч. 2 : Анализ и синтез систем / Ю. А. Ершов, С. И. Щукин. – 2-е изд., испр. и доп. – Москва : Юрайт, 2017. – 348 с.

4 **Кореневский, Н. А.** Проектирование биотехнических систем медицинского назначения. Средства оценки состояния биообъектов : учебник / Н. А. Кореневский, З. М. Юлдашев. – Старый Оскол : ТНТ, 2017. – 456 с.

5 **Кузин, А. В.** Базы данных : учебное пособие / А. В. Кузин, С. В. Левонисова. – 6-е изд., стер. – Москва : Академия, 2016. – 320 с.

6 **Леоненков, А. В.** Самоучитель UML 2 / А. В. Леоненков. – Санкт-Петербург : БХВ-Петербург, 2007. – 576 с.

7 **Рудинский, И. Д.** Технология проектирования автоматизированных систем обработки информации и управления : учебное пособие для вузов. – Москва : Горячая линия-Телеком, 2011. – 304 с.

8 **Орлов, С. А.** Программная инженерия. Технологии разработки программного обеспечения : учебник / С. А. Орлов. – 5-е изд., обновл. и доп. – Санкт-Петербург : Питер, 2016. – 640 с. : ил.

9 **Шустова, Л. И.** Базы данных : учебник / Л. И. Шустова, О. В. Тараканов. – Москва : ИНФРА-М, 2017. – 304 с.

10 Автоматизация обработки биомедицинской информации : методические рекомендации к практическим занятиям для студентов направления подготовки 12.03.04 «Биотехнические системы и технологии» дневной формы обучения / Сост. К. В. Захарченков, Т. В. Мрочек. – Могилев : Белорус.-Рос. ун-т, 2018. – 32 с.

Электронная библиотека Белорусско-Российского университета
http://e.biblio.bru.by/