ТЕМА 3

План лекции:

1. Основные и дополнительные элементы объектно-ориентированного подхода.
2. Унифицированный язык моделирования UML.
3. Средства автоматизированной разработки Rational Rose и Rational XDE.

3.1 Основные и дополнительные элементы объектно-ориентированного подхода

Главное отличие объектного подхода от структурного заключается в объектной декомпозиции системы. При этом статическая структура системы описывается в терминах объектов и связей между ними, а поведение системы – в терминах обмена сообщениями между объектами. Концептуальной основой объектного подхода является объектная модель, главными элементами которой считаются: абстрагирование, инкапсуляция, модульность, иерархия. Кроме того, имеются три дополнительных элемента: типизация, параллелизм, устойчивость. Рассмотрим все элементы с точки зрения создания ПО с помощью CASE-средств.

Абстрагирование – это выделение существенных характеристик некоторого объекта, которые отличают его от всех других видов объектов и, таким образом, четко определяют его концептуальные границы относительно дальнейшего рассмотрения и анализа. Абстрагирование позволяет отделить существенные особенности поведения объекта от деталей их реализации. Выбор правильного набора абстракций – это главная задача объектно-ориентированного подхода.

Инкапсуляция – это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение. Инкапсуляция служит для того, чтобы разделить интерфейс и внутреннюю реализацию объекта.

Модульность – это свойство системы, связанное с ее декомпозицией на ряд внутренне сильно связанных, но слабо связанных между собой модулей. Инкапсуляция и модульность создают барьеры между абстракциями.

Иерархия – это ранжированная или упорядоченная система абстракций. Иерархия по номенклатуре – это структура классов, а иерархия по составу – это структура объектов.

Типизация – это ограничение, накладываемое на класс объектов и препятствующее взаимозаменяемости различных классов.

Параллелизм – это свойство объектов находиться в активном или пассивном состоянии и различать между собой активные и пассивные объекты.

Устойчивость – это свойство объекта существовать во времени и в пространстве вне зависимости от процесса, породившего данный объект.

Еще два важных понятия объектно-ориентированного подхода – это полиморфизм и наследование. *Полиморфизм* можно интерпретировать, как способность класса принадлежать более чем одному типу. *Наследование* означает построение новых классов, на основе существующих с возможностью добавления

или переопределения данных и методов. Система изначально строится с учетом ее эволюции, которую позволяют реализовать полиморфизм и наследование: потомки могут добавлять в родительские классы новые структуры данных и методы.

Благодаря применению абстрагирования, модульности и полиморфизма на всех стадиях разработки ПС существует согласованность между моделями всех этапов ЖЦ, когда модели ранних стадий могут быть сравнены с моделями реализации.

3.2 Унифицированный язык моделирования UML

Методы объектно-ориентированного анализа и проектирования включают в себя язык моделирования и описание процессов моделирования. Язык моделирования UML (Unified Modeling Language) – это нотация, которая используется методом для описания проектов. *Нотация* представляет собой совокупность графических объектов, которые используются в моделях. Она является синтаксисом языка моделирования. *Процесс* – это описание шагов, которые необходимо выполнить при разработке проекта.

Авторами UML являются основоположники объектно-ориентированного подхода: Буч, Рамбо, Якобсон. UML объединяет в себе методы объектного подхода, дополняя их новыми возможностями. UML не привязан к какой-либо конкретной методологии или ЖЦ и может использоваться со всеми существующими методологиями.

Создание UML началось в 1994 году, а в 1995 появилась первая спецификация языка. В 2000 году появилась версия UML 1.4 как существенное расширение UML, достигнутое добавлением семантики действий. Это было серьезным достижением, поскольку сделало спецификацию UML полной в вычислительном отношении, что обеспечило возможность создавать UML-модели исполняемыми. В 2005 году была завершена спецификация UML 2.0.

Основная идея UML – возможность моделировать ПО и другие системы как наборы взаимодействующих объектов. В UML-модели есть два аспекта:

- *статическая структура* – описывает, какие типы объектов важны для моделирования системы и как они взаимосвязаны;

- *динамическое поведение* – описывает ЖЦ этих объектов и то, как они взаимодействуют друг с другом для обеспечения требуемой функциональности системы.

Основные цели создания унифицированного языка моделирования:

1. Предоставить пользователям готовый к применению выразительный язык визуального моделирования, позволяющий разрабатывать осмысленные модели и обмениваться ими.

2. Предусмотреть механизмы для расширения базовых концепций.

3. Обеспечить независимость UML от конкретных языков программирования и процессов разработки.

4. Создать формальную основу для понимания языка моделирования.

5. Стимулировать рост рынка объектно-ориентированных инструментальных средств.

6. Интегрировать лучший практический опыт.

Семантика языка UML представляет собой некоторую метамодель, которая определяет абстрактный синтаксис и семантику понятий объектного моделирования на языке UML. Семантика определяется для двух видов объектных моделей: структурных моделей и моделей поведения. Структурные модели, известные также как статические модели, описывают структуру сущностей или компонентов некоторой системы, включая их классы, интерфейсы, атрибуты и отношения. Модели поведения, называемые иногда динамическими моделями, описывают поведение или функционирование объектов системы, включая их методы, взаимодействие и сотрудничество между ними, а также процесс изменения состояний отдельных компонентов и системы в целом.

Для решения столь широкого диапазона задач моделирования разработана достаточно полная семантика для всех компонентов графической нотации. Требования семантики языка UML конкретизируются при построении отдельных видов диаграмм.

В настоящее время UML принят в качестве стандартного языка моделирования и получил широкую поддержку в индустрии ПО. UML взят на вооружение самыми известными производителями ПО: IBM, Microsoft, Hewlett-Packard, Oracle. Большинство современных CASE-средств разрабатывается на основе UML.

3.3 Средства автоматизированной разработки Rational Rose и Rational XDE

Рассмотрим несколько наиболее популярных CASE-средств, созднанных на основе объектно-ориентированного подхода и использующих нотацию UML.

Одним из таких инструментом является CASE-средство *Rational Rose*.

Rational Rose – это объектно-ориентированное средство автоматизированного проектирования ПС. В его основе лежит CASE-технология, комплексный подход и использование единой унифицированной нотации на всех этапах жизненного цикла создания ПС.

Графические возможности продукта позволяют решать задачи, связанные с проектированием, на различных уровнях абстракции: от общей модели процессов предприятия до конкретной модели класса в создаваемом ПО. В среде *Rational Rose* проектировщик и программист работают в тандеме. Первый создает логическую модель системы, а второй дополняет ее моделями классов на конкретном языке программирования. В настоящее время продукт обеспечивает генерацию кода по модели на ряде языков программирования: *Microsoft Visual C++, Ada, Java, Visual Basic, CORBA, XML, COM, Oracle*. Кроме того, разрабатываются специальные мосты к не входящим в стандартную поставку языкам, например к *Delphi*.

Rational Rose поддерживает проектирование, основанное на двух способах: прямом и обратном. В первом режиме разработчик строит диаграммы классов и их взаимодействия, а на выходе получает сгенерированный код. Во втором режиме возможно построение модели на базе имеющегося исходного кода. Отсюда следует главная возможность для разработчика: повторное проектирование. Программист описывает классы в *Rational Rose*, генерирует код, вносит

изменения в модель и снова пропускает ее через *Rational Rose* для получения обновленного результата.

Модели в виде *UML*-диаграмм, созданные в среде *Rational Rose*, имеют целый ряд замечательных особенностей. Они удобны для понимания алгоритмов работы, взаимосвязей между объектами системы и ее поведения в целом, а также позволяют непосредственно из проекта автоматически построить исходный код. Можно выделить ряд преимуществ, получаемых от применения *Rational Rose*:

- сокращение цикла разработки приложения “заказчик-программист-заказчик”;
- увеличение продуктивности работы программиста;
- улучшение потребительских качеств создаваемых программ за счет ориентации на пользователей и бизнес;
- способность вести большие проекты и группы проектов;
- возможность повторного использования уже созданного программного обеспечения за счет упора на разбор их архитектуры и компонентов.

После запуска системы открывается ее главное окно, показанное на рис. 3.1. В верхней части экрана находится меню и стандартная панель инструментов. Она видна всегда, и ее кнопки соответствуют командам, которые могут использоваться для работы с любой диаграммой.

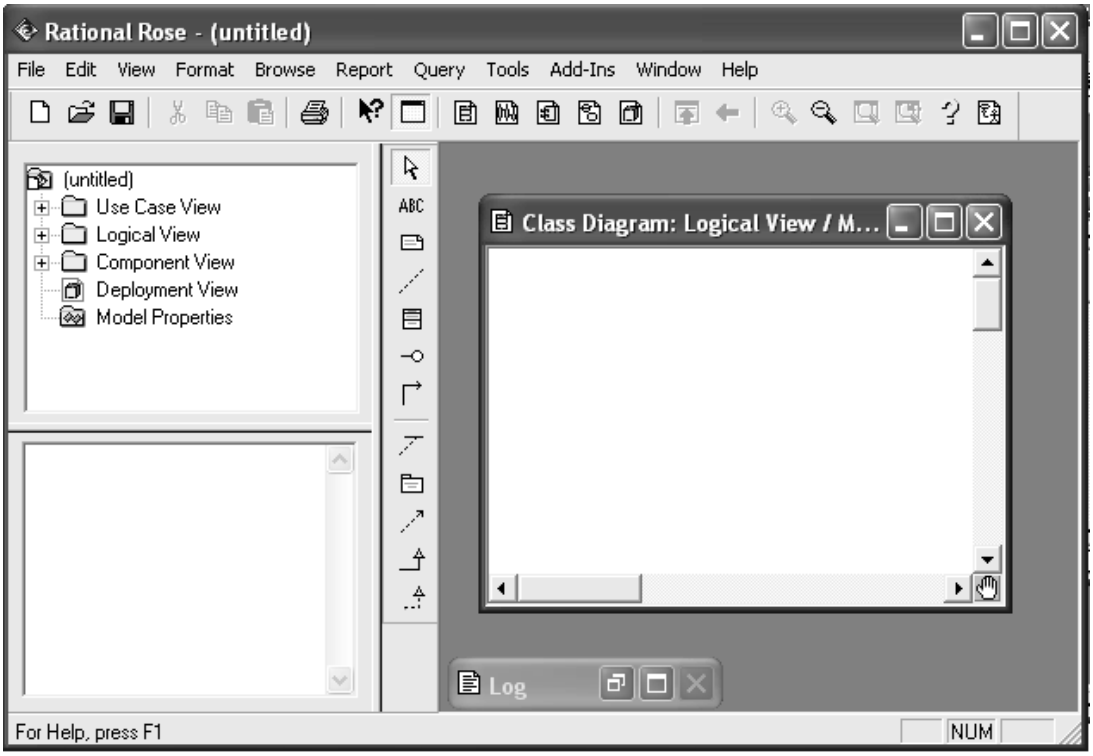

Рисунок 3.1 – Главное окно *Rational Rose*

Слева расположено окно *Browser*, представляющее собой иерархическую структуру и предназначенное для выполнения ряда действий:

- просмотр и добавление элементов к модели;
- просмотр существующих отношений между элементами модели;
- перемещение и переименование элементов модели;

- добавление элементов модели к диаграмме;
- связывание элемента с файлом или адресом Интернета;
- группирование элементов в пакеты;
- работа с детализированной спецификацией элемента;
- открытие диаграммы.

Каждый объект в *Rational Rose* имеет свое контекстное меню, посредством которого изменяются свойства и выполняются действия над объектом. Под *Browser* находится окно *Documentation*, предназначенное для документирования элементов модели *Rational Rose*. При описании класса вся информация из этого окна появится затем как комментарий в сгенерированном коде и в отчетах, создаваемых в среде *Rational Rose*. В случае смены активного элемента содержание *Documentation* автоматически обновляется.

В правой части экрана, называемой рабочим столом *Rational Rose*, находятся открытые в данный момент диаграммы. При создании новой модели на рабочем столе открывается *Class Diagram* (диаграмма классов). Окна *Browser* и *Diagram* разделены строкой инструментов, которая изменяется в зависимости от типа активной диаграммы. Внизу рабочего стола видно свернутое окно *Log* (протокол), в котором фиксируются все действия, выполненные над диаграммами, также туда попадают сообщения об ошибках, произошедших в течение работы. Информация заносится в окно *Log* независимо от того, свернуто оно или вообще закрыто.

При изменении диаграмм в области рабочего стола *Rational Rose* автоматически обновит структуру *Browser*. Аналогично при внесении изменений в элемент с помощью *Browser Rational Rose* автоматически обновит соответствующие диаграммы. Это помогает поддерживать модель в непротиворечивом состоянии.

Рассмотрим состав и назначение пунктов меню главного окна *Rational Rose*, приведенного на рис. 3.1.

- *File* используется для сохранения, загрузки, обновления проекта, печати диаграмм и дополнительных настроек.
- *Edit* предназначен для копирования и восстановления данных, а также для редактирования свойств и стилей объектов.
- *View* применяется для настройки представления окон меню и строк инструментов.
- *Format* позволяет изменять параметры отображения объектов, такие как шрифт, заливку, формат линий и т.д.
- *Browse* предназначен для навигации между диаграммами и спецификациями диаграмм, представленных в модели.
- *Report* предназначен для получения различного вида справок и отчетов.
- *Query* предоставляет возможности контролировать, какие элементы модели будут показаны на текущей диаграмме, и выполнять различные манипуляции с объектами диаграмм: скрывать, добавлять, фильтровать.
- *Tools* предоставляет доступ к различным дополнительным инструментам и подключаемым модулям.

- *Add-Ins* предоставляет доступ к менеджеру подключаемых модулей.
- *Window* позволяет управлять окнами на рабочем столе.
- *Help* позволяет получать справочную информацию.

Все перечисленные возможности среды *Rational Rose* служат для моделирования прикладной программной системы в виде совокупности диаграмм на основе графических средств языка *UML*.

Еще одним лидером на мировом рынке CASE-продуктов является программное средство *Rational XDE*.

Rational XDE – это расширенная среда разработки (*eXtended Development Environment*), полностью интегрируемая в *Microsoft Visual Studio.NET*. *Rational XDE* – это средство, позволяющее проектировать программную систему при помощи *UML* моделей. Среда позволяет создавать диаграммы и генерировать по полученным моделям исходный код приложения, а также проводить обратное проектирование, то есть строить диаграммы по разработанному ранее исходному коду.

Главное отличие *Rational XDE* от своего предшественника *Rational Rose* – это полная интеграция с платформой *Microsoft Visual Studio.NET*, позволяющая в одной оболочке работать как с моделями создаваемой системы, так и непосредственно с кодом. В то же время в *Rational XDE* сохранились принципы работы *Rational Rose*, относящиеся к созданию диаграмм. В результате интеграции с *Microsoft Visual Studio.NET* *Rational XDE* потеряла часть своей универсальности по сравнению с Rational Rose, касающуюся в основном возможности генерации программного кода практически для любых языков программирования. В *Rational XDE для .NET* возможна синхронизация модели и кода только для языков, которые поддерживаются *Microsoft Visual Studio.NET*. На данный момент это *Visual Basic, C#*. При этом кроме *UML* диаграмм *XDE* позволяет создавать *Free Form* (свободные формы), в которых не отслеживается нотация *UML*, и которые могут включать в себя значки из различных *UML* диаграмм, что не допускалась в *Rational Rose*. В *Rational XDE* включены дополнительные фигуры и значки, которые позволяют аналитику нагляднее отображать в создаваемых моделях реальное положение дел.

Язык C#, на котором генерируется код в среде *Rational XDE*, разработан компанией Microsoft для платформы *.NET*. Она представляет собой обширную библиотеку классов, инфраструктуру и инструментальные средства для создания межплатформенных, не зависящих от языка программирования приложений. На платформе *.NET* создана *ASP.NET* – технология активных серверных страниц. В приложении *ASP.NET* доступна вся библиотека *.NET* Framework, значительно ускоряющая и облегчающая разработку сложных сетевых программных систем.

На рис. 3.2 показано главное окно программы *Rational XDE*. На его внешнем виде отразилась интеграция с *Microsoft Visual Studio.NET*. В результате рабочий стол *Visual Studio.NET* изменился и на него добавились окна, отвечающие за моделирование программной системы. В центре экрана расположено окно документов, в котором можно открывать код, ресурсы и диаграммы *UML*,

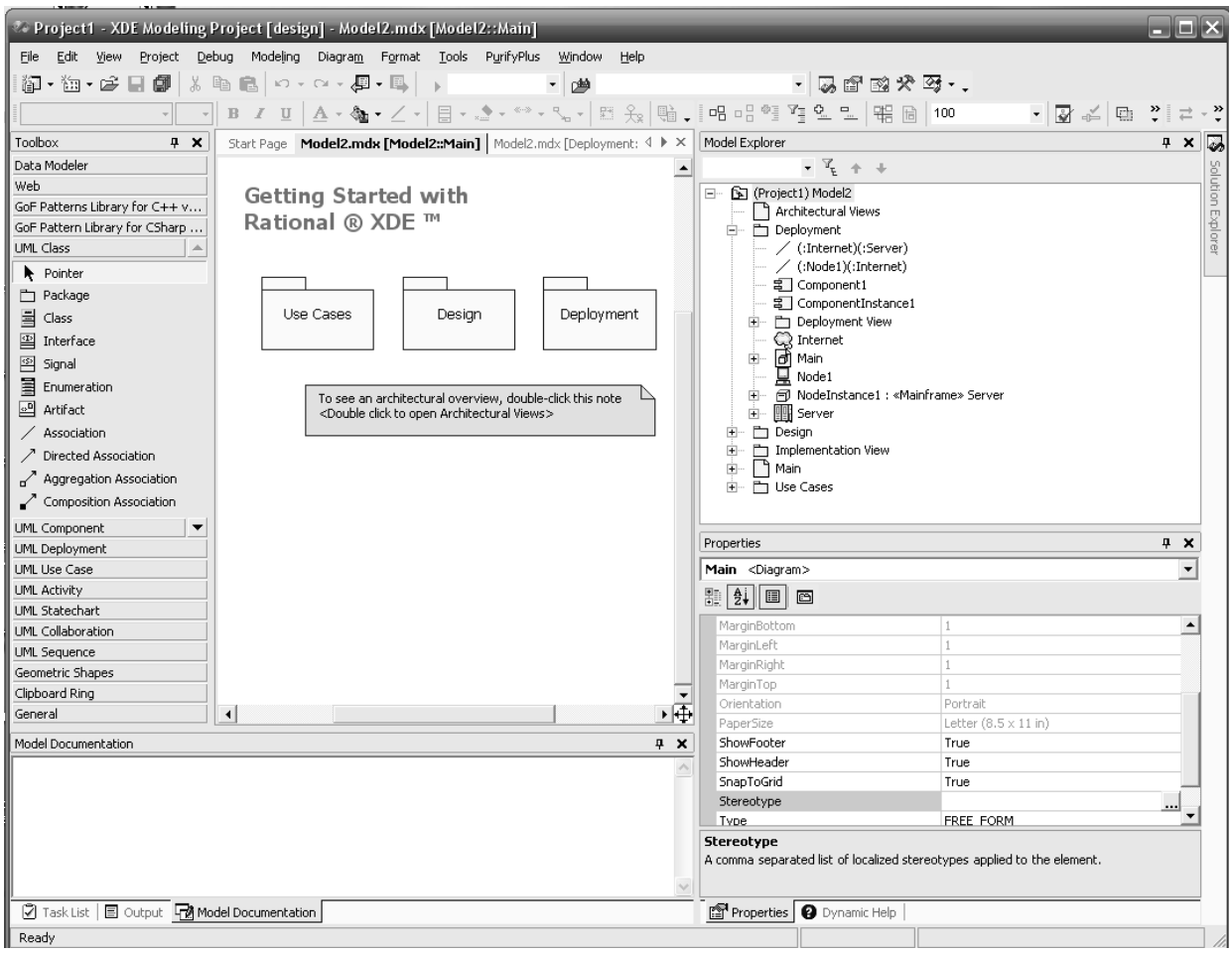

Рисунок 3.2 – Главное окно Rational XDE

создаваемые в модели. Справа в окне *Explorer* добавилась закладка Model Explorer, позволяющая перемещаться по модели. Под ней – окно *XDE Code Properties*, которое показывает свойства выбранной диаграммы. Внизу – закладка *Model Documentation*, отражающая документацию модели. Слева, в окне Toolbox, добавилось большое количество инструментов, необходимых для работы с *UML*- моделями в отдельных разделах, которые появляются в момент активизации одной из диаграмм.

Основная работа с элементами модели осуществляется при помощи окон *Model Explorer* и рабочего стола диаграммы. При этом за перемещение по модели отвечает *Model Explorer*, а редактирование лучше выполнять на рабочем столе. Одним из отличий *Rational XDE* от предыдущей версии явилось использование окна Toolbox, содержащего дополнительные инструменты для работы над проектом. Строки инструментов (Toolbar) также остались, но их роль сократилась до управления основными режимами, а создание новых элементов теперь производится с использованием *Toolbox*.

Model Explorer позволяет осуществлять навигацию по элементам модели, представленным в иерархическом виде. Из контекстного меню можно добавлять, удалять и изменять как диаграммы *UML*, так и элементы диаграмм, синхронизировать исходный код и диаграммы, работать с шаблонами кода, т.е. полностью управлять созданием модели программной системы. *Model Explorer* представляет собой аналог стандартного обозревателя *Windows*.

В окне *Model Documentation* отображается текст документации, относящийся к модели, диаграмме или выделенному элементу на ней. Окно документации представляет собой текстовый редактор, позволяющий описывать цели создания, поведение и другую информацию. Документация в *Rational XDE* обновляется вместе с моделью. В случае если разработчик внес в исходный код комментарий, а после этого модель обновилась, то все комментарии сразу же будут отображены в окне документации.

Окно Toolbox сменило строку инструментов *Toolbar*. С помощью Toolbox выполняются основные функции по работе с элементами на диаграмме.

Аналогично тому, как это происходило в *Rational Rose*, в среде *Rational XDE* все действия над объектами выполняются посредством контекстного меню. Оно зависит от набора функций, доступных для применения к конкретному объекту. При установке *Rational XDE* добавляет свои пункты в главное меню *Visual Studio .NET*, изменяет некоторые пункты, установленные по умолчанию, а также добавляет свои строки инструментов. Это связано с тем, что после установки *Rational XDE* в среде *Visual Studio .NET* можно работать с графическими диаграммами, а не только с программным кодом и ресурсами проекта. Рассмотрим пункты меню, добавленные *Rational XDE*, и предоставляемые ими возможности по работе с моделями или их элементами.

Modeling. Этот раздел меню предназначен для добавления диаграмм и их элементов в проект. Пункты моделирования позволяют добавлять в проект UML-элементы или диаграммы; проверять корректность диаграмм; проверять из проекта ссылки на другие модели; исправлять ошибки во внешних ссылках; устанавливать пути доступа к файлам модели; использовать шаблоны при построении модели.

Diagram. Этот раздел меню предназначен для общего управления значками на диаграммах. Его пункты позволяют: автоматически расставить значки на текущей диаграмме; изменить на диаграмме положение выделенного элемента; переместить фокус просмотра на выделенный элемент диаграммы; добавить на диаграмму элементы, связанные с выделенным; активизировать окно настройки визуализации соединений; работать с надписями на стрелках соединителях; управлять перерисовкой диаграммы и найти отмеченный на ней элемент.

Format. Этот раздел меню позволяет управлять форматированием диаграммы. Его пункты позволяют показать или скрыть список атрибутов, операций и сигналов в классе; показать или скрыть сигнатуры операций или сигналов; изменять стиль визуализации элементов диаграмм и их стереотипов; изменять стиль выделенной линии; поддерживать заданный размер элемента диаграммы; включить или отключить показ имени родительского контейнера для выделенного элемента; установить одинаковые настройки для всех выделенных элементов диаграммы.