Лабораторная работа № 2

Генерация кода C++ на основе модели UML

Цель работы:
Изучение возможностей языка UML
Построение диаграммы классов
Построение диаграммы компонентов
Генерация кода

Приводится подробное руководство по генерации кода на языке программирования **C++** средствами **Rational Rose.** Соответствующий генератор кода не включается по умолчанию — следует выбрать элемент меню **Add-lns →Add In Manager**, в одноименном диалоговом окне установить флажок **Rose C++** и закрыть окно щелчком на кнопке **ОК**.

Ниже перечислены фазы процессов генерации кода и обратного восстановления модели.

Генерация кода

1. Создание наборов свойств.
2. Определение спецификаций компонентов.
3. Выбор языка **C++** для компонентов.
4. Отнесение классов к компонентам.
5. Связывание наборов свойств с элементами модели.
6. Генерация кода.
7. Анализ ошибок.

Генерация кода

1. Создание наборов свойств

При генерации кода учитываются свойства проекта в целом, а также свойства уровней классов, ролей, атрибутов и операций. К свойствам, регламентирующим характеристики проекта как такового, относятся имя файла проекта, названия контейнерных классов, используемых по умолчанию, и местоположение генерируемого кода. Свойства уровня класса обусловливают необходимость и способы создания конструкторов, деструкторов, конструкторов копии, операторов сравнения и методов **get/set**. Набор свойств роли определяет потребность в использовании методов **get/set**, признаки видимости методов и варианты применения того или иного контейнерного класса. Свойства операции позволяют отнести последнюю к одной из разновидностей (**common** — общая, **virtual** — виртуальная, **abstract** — абстрактная, **static** — статическая, **friend** —"дружественная") и/или придать ей статус "постоянной" (**constant**). Rational Rose предоставляет возможность создания любого количества наборов свойств, отвечающих существу проекта, и их редактирования. Для каждого класса генерируются два файла — файл заголовка **(.h**) и файл спецификации **(.cpp**).

При работе над типичным проектом обязанности по формированию наборов свойств генерируемого кода распределяются между несколькими сотрудниками, а результаты используются всеми участниками группы. Вот некоторые примеры часто создаваемых наборов свойств: "виртуальный деструктор", "виртуальная операция", "абстрактная операция", "статическая операция".

Как создать набор свойств

1. Выбрать элемент меню **Tools→Options**.
2. Перейти на вкладку **C++** диалогового окна **Options**.
3. В раскрывающемся списке **Type** установить требуемый тип набора свойств.

4. Щелкнуть на кнопке **Clone**, чтобы открыть диалоговое окно **Clone Property Set**.
5. Ввести наименование нового набора свойств и закрыть окно щелчком на кнопке **ОК**.
6. В списке **Model Properties** выбрать свойство, подлежащее модификации, и щелкнуть в пределах столбца **Value**.
7. Ввести новое значение свойства либо выбрать таковое с помощью раскрывающегося списка.
8. Повторить действия, перечисленные в п. п. 6, 7 для каждого свойства, которое должно быть изменено.
9. Щелкнуть на кнопке **Apply**, чтобы сохранить информацию.
10. Повторить действия, перечисленные в п.п. 3-9, с целью создания остальных наборов свойств.
11. Закрыть диалоговое окно **Options** щелчком на кнопке **ОК**.

Процесс создания набора свойств «**Виртуальный деструктор**» показан на рисунке 1

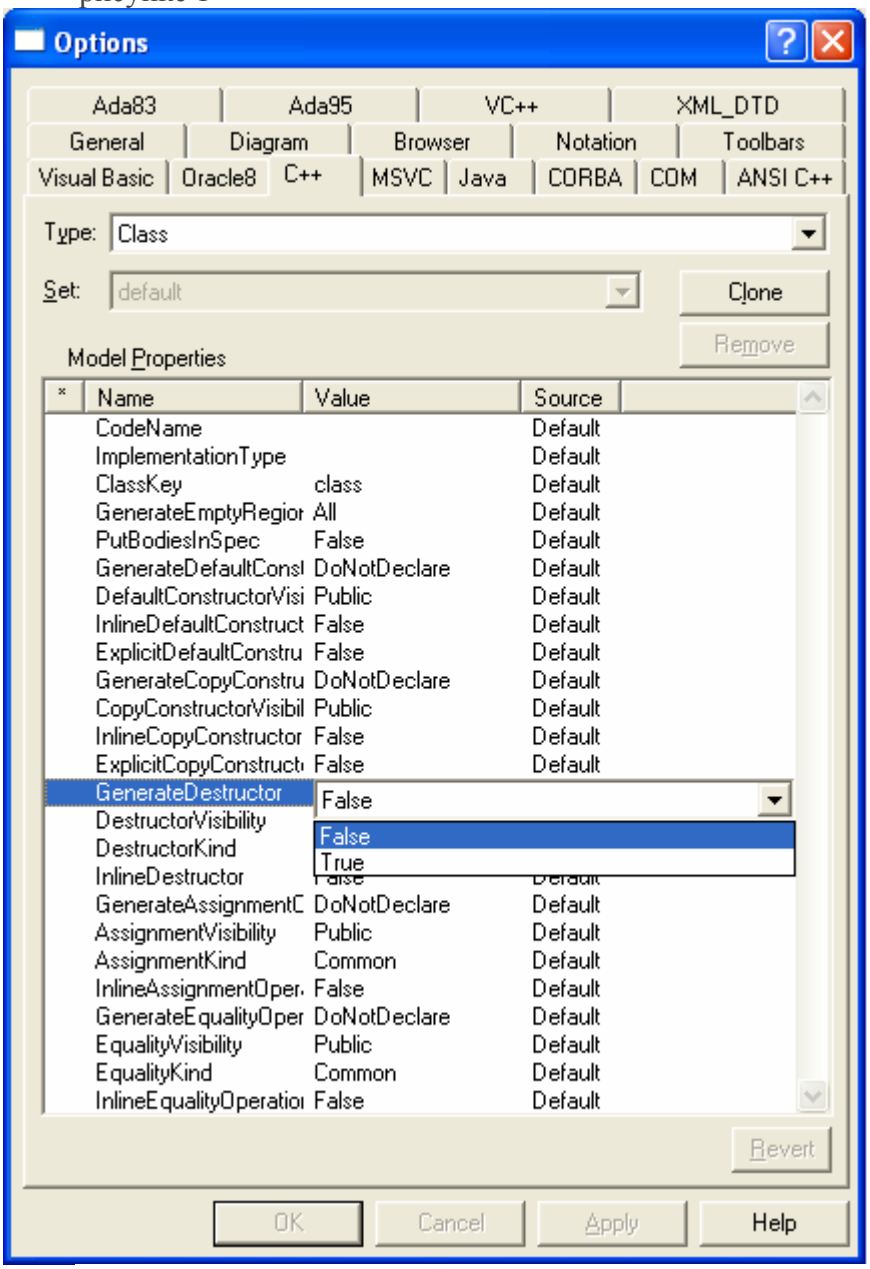

Рисунок 1. Так создается набор свойств

2. Определение спецификаций компонентов

Rational Rose генерирует код, принимая во внимание номенклатуру созданных компонентов в совокупности с их стереотипами. Для каждого компонента без стереотипа система генерирует файл **.h**, содержащий информацию объявления и определения соответствующего класса. Если компонент снабжен стереотипом **Package Specification**, генерируется файл **.h** с объявлением класса. Если же при этом существует надлежащий компонент со стереотипом **Package Body**, генерируется и файл **.cpp** с определением класса.

Как определить или создать стереотип компонента

1. Двойным щелчком на элементе дерева в окне Browser, представляющем диаграмму компонентов, открыть окно диаграммы.
2. Расположить курсор мыши над элементом диаграммы, отвечающим требуемому компоненту, и щелкнуть правой кнопкой, чтобы активизировать контекстное меню.
3. Выбрать элемент меню **Open Specification**.
4. Перейти на вкладку **General** диалогового окна **Component Specification**.
5. В поле **Stereotype** ввести значение стереотипа либо выбрать таковое с помощью раскрывающегося списка.
6. Закрыть диалоговое окно щелчком на кнопке **ОК**.

Диалоговое окно **Component Specification** изображено на рис..2.

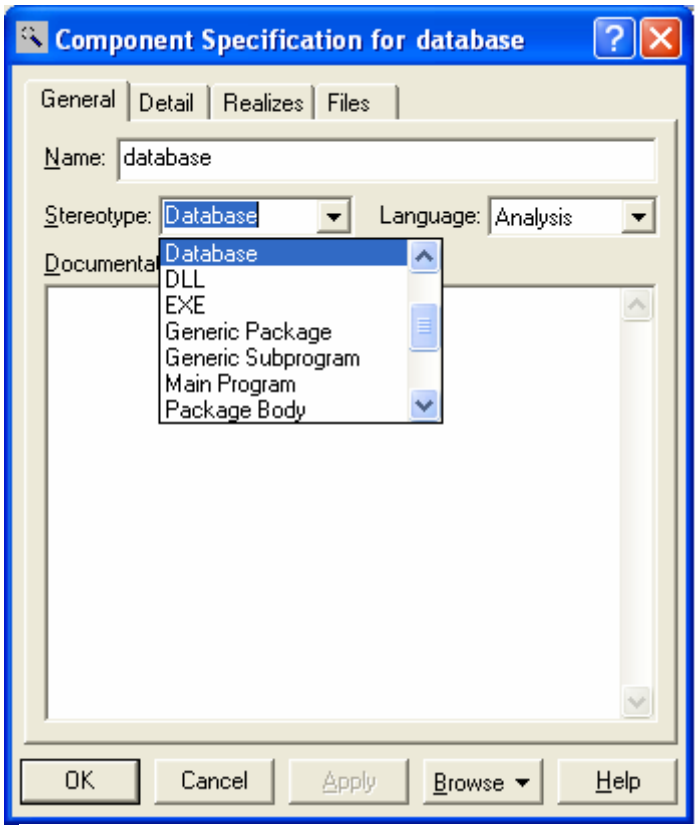

Рисунок 2. Спецификация компонента

Как создать заголовок и тело компонента

1. Двойным щелчком на элементе дерева в окне **Browser**, представляющем диаграмму компонентов, открыть окно диаграммы.
2. Расположить курсор мыши над элементом диаграммы, отвечающим требуемому компоненту, и щелкнуть правой кнопкой, чтобы активизировать

контекстное меню.

3. Выбрать элемент меню **Open Specification**.
4. Перейти на вкладку **General** диалогового окна **Component Specification**.
5. В раскрывающемся списке **Stereotype** выбрать значение стереотипа **Package Specification** для файла заголовка компонента либо значение **Package Body** — для файла, содержащего тело кода компонента.
6. Закрыть диалоговое окно щелчком на кнопке **ОК**.

Пример диаграммы компонентов, элементы которой отвечают файлам **.h** и **.cpp**, приведен на рис. 3. Светлый компонент соответствует файлу заголовка, темный - файлу тела кода.

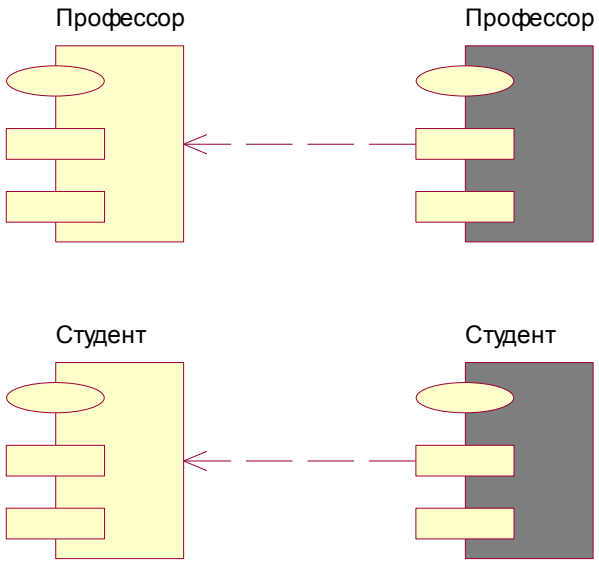

Рис. 3. Уточненная диаграмма компонентов

3. Выбор языка C++ для компонентов

Как только компоненты, представляющие файлы **.h** и **.cpp**, созданы, им должен быть поставлен в соответствие язык программирования (C++). (Если языком, предлагаемым по умолчанию, является C++ - обратитесь к раскрывающемуся списку Default Language на вкладке Notation диалогового окна Options, активизируемого командой меню Tools'→ Options, - система автоматически выбирает опцию C++ для каждого компонента модели.)

Как выбрать язык программирования для компонента

1. Щелчком правой кнопки мыши указать компонент в дереве окна **Browser** либо на диаграмме компонентов и активизировать контекстное меню.
2. Выбрать элемент меню **Open Specification**.
3. Перейти на вкладку **General** диалогового окна **Component Specification**.
4. В раскрывающемся списке **Language** выбрать требуемую опцию (в данном случае - **C++).**
5. Закрыть диалоговое окно щелчком на кнопке **ОК**.
6.

Окно спецификации компонента "курс" показано на рис. .4.

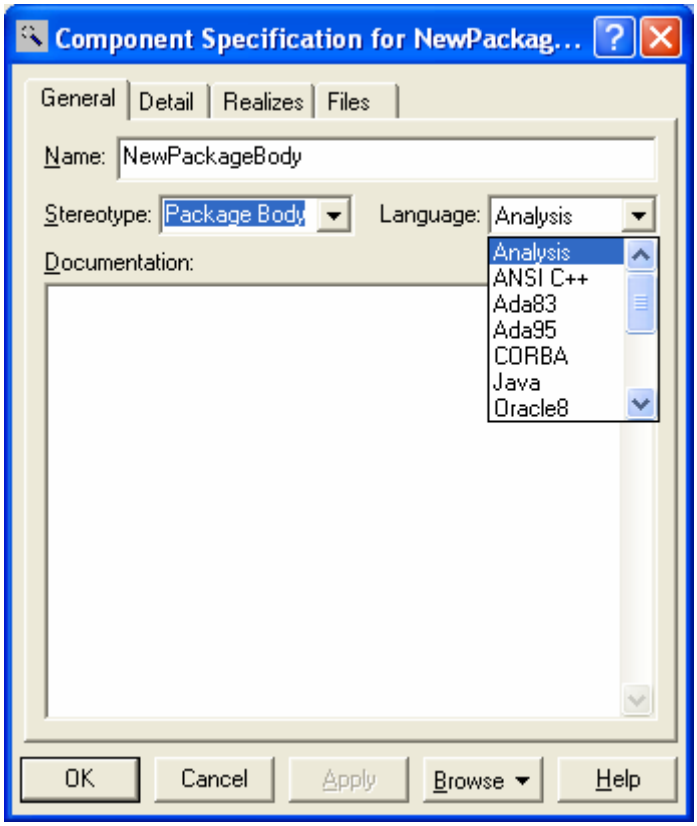

Рис. 4. Так выбирается язык программирования для компонента

4. Отнесение классов к компонентам

После создания компонентов, представляющих файлы .h, с ними следует сопоставить те или иные классы модели.

Как отнести класс к компоненту

1. Двойным щелчком на элементе дерева в окне Browser, представляющем диаграмму с компонентами файлов .h и .cpp, открыть окно диаграммы.
2. Выбрать класс в дереве окна Browser, перетащить в окно диаграммы и опустить на компонент, соответствующий требуемому файлу .h.

5. Связывание наборов свойств с элементами модели

Каждый элемент модели (например, класс, атрибут или роль) анализируется системой с целью выявления свойств, которыми должен обладать генерируемый код. Если элемент должен обладать свойствами, отличными от тех, которые предусмотрены в наборе, предлагаемом по умолчанию, с элементом связывается тот или иной созданный набор свойств.

Как связать набор свойств с элементом модели

Щелчком правой кнопки мыши указать элемент модели в дереве окна Browser либо на диаграмме и активизировать контекстное меню.

1. Выбрать элемент меню **Open Specification**.
2. Перейти на вкладку **C++** диалогового окна спецификации элемента.
3. В раскрывающемся списке **Set** выбрать требуемый набор свойств.
4. Закрыть диалоговое окно щелчком на кнопке **ОК**.

Поскольку адекватного набора свойств, подходящего для каждой комбинации

элементов, заведомо не существует, при рассмотрении определенного элемента то или иное свойство можно переопределить — даже в том случае, если свойство входит в набор свойств, предлагаемый по умолчанию.

Как переопределить свойство элемента модели

1. Щелчком правой кнопки мыши указать элемент модели в дереве окна **Browser** либо на диаграмме и активизировать контекстное меню.
2. Выбрать элемент меню **Open Specification**.
3. Перейти на вкладку **C++** диалогового окна спецификации элемента
4. В раскрывающемся списке Set выбрать требуемый набор свойств.
5. В списке **Model Properties** указать свойство, подлежащее модификации, и щелкнуть в пределах столбца **Value**.
6. Ввести новое значение свойства либо выбрать таковое с помощью раскрывающегося списка.
7. Повторить действия, перечисленные в п.п. 5, *6,* для каждого свойства, которое должно быть изменено.
8. Щелкнуть на кнопке **Override**.
9. Закрыть диалоговое окно щелчком на кнопке **ОК**.

6. Генерация кода

Код может быть сгенерирован для пакета в целом, для отдельного компонента либо группы компонентов. В качестве имени файла, в который помещается код, выбирается наименование пакета или компонента. Файл располагается в структуре каталогов, соответствующей поддереву Component View дерева Browser.

Как сгенерировать код

1. Щелчком выбрать пакет, компонент или группу компонентов в дереве окна **Browser** либо на диаграмме.
2. Выбрать элемент меню **Tools → C++ → Code Generation**.
3. Система осуществит генерацию кода и воспроизведет информацию о результатах в диалоговом окне **Code Generation Status**.

Окно **Code Generation Status** с данными об итогах генерации приведено на рис. А.6 (символы кириллицы в названиях элементов модели отображаются посредством восьмеричных кодов. — *Прим, перев.).*

7. Анализ ошибок

Предупреждающие сообщения и информация об ошибках выводятся в окно протокола (**Log Window**, или **Output Window**). (Чтобы открыть окно протокола, достаточно выбрать элемент меню **View → Log**.) Если дизайн класса не завершен, система отобразит в окне предупреждающее сообщение. Подобная ситуация возникает в процессе итеративной разработки, когда классы не всегда реализуются в пределах одной отдельно взятой версии. Ниже перечислено несколько типичных сообщений об ошибках и предупреждениях, выводимых системой по мере генерации кода.

- **Error: Missing attribute data type. Void is assumed**. (Ошибка: отсутствует тип данных атрибута; подразумевается тип **void**.)
- **Warning**: **Unspecified multiplicity/cardinality indicators. One is assumed**. (Предупреждение: не заданы признаки множественности; подразумевается значение "один".)
- **Warning: Missing operation return type. Void is assumed**. (Предупреждение: отсутствует тип значения, возвращаемого операцией; подразумевается тип **void**.)

Окно Output Window показано на рис. 5.

116:31:42j Generating code to "d:\Rational\Rose\C++\source"
!16:31:42| ::: Module Body\317A356\353\374\347\356\342\340\362\345\353\374(.cpp)
11631:42| — 0 code regions found In previous version of code.
Il6:31:42| — Class specification \317\356\353\374\347\356\342\34<ЛЭ62\345\353\374 (XXXXXXXXXXXX) l)6:31:42|
Error »•« Class attribute (XXX)" with unspecified type; void Is assumed Il6:31:42i
Error. •»• Class attribute "\5\360 (XXXXXXXXXXXXX)" with" 116:31:42)
Error «•« Attribute "\350\354\377 (XXX)" with unspecified type: void is assumed 16:31:42]
Error:««««Attribute \35ба54Ш5\360 (XXXXXXXXXXXXX)"with unsp I16:31:42
| — Class XXXXXXXXXXX (lnlines)
116:31:42) Error »«» Class attribute "\350V354\377 (XXX)" with unspecified type; void Is assumed
Il6:31:42| Error. »•» Class attribute
36Ш50\36Ш62\345\354a55\Э73\Э51\315\Э56\354\Э45\360 (XXXXXXXXXXXXX)" with-
Il6:31:42|
- **Class body \317«56\353a74\347U56\342V340\362a45\353\374 (XXXXXXXXXXX)**
-

Рис.5. Так выглядит окно протокола работы Rational Rose

Выполните лабораторную работу в соответствии с вариантом.

1. Пассажир бронирует билет на рейс у агента. Классы: **пассажир с атрибутами**: Имя, фамилия, адрес, №паспорта, город вылета, город прилета; **с операциями**: заказать, купить
2. **Агент с атрибутами**, Фамилия, номер агента, **с операциями**: бронировать, продать

3. Клиент сдает автомобиль в автосервис. Классы: **Клиент с атрибутами**: Фамилия, марка машины, пробег, неисправность; с операциями: сдать в ремонт, взять из ремонта , **приемщик с атрибутами**: фамилия, дата приема, дата выдачи, с операциями: принять машину, выдать машину; **слесарь с атрибутами**: фамилия, специализация,

4. Покупатель покупает книгу в книжном магазине. Актеры: **покупатель с атрибутами:** специальность, интерес, **продавец с атрибутами**, **кладовщик с атрибутами**
5. Клиент берет видеокассету в пункте проката. Актеры: **клиент с атрибутами**, **прокатчик с атрибутами**

6. Пассажир приходит на регистрацию рейса в аэропорт. Актеры: **пассажир с атрибутами:** фамилия, дата вылета, город прилета, **агент с атрибутами**, **приемщик багажа с атрибутами**

Отчет должен содержать:

1) Название лабораторной работы;
2) Цель работы;
3) Вариант задания;
4) Распечатки диаграмм и программных кодов.