Лабораторная работа № 8
Диаграмма компонентов

Назначение диаграммы

Component Diagram (диаграмма компонентов) позволяет создать физическое отражение текущей модели. Диаграмма компонентов показывает организацию и взаимосвязи программных компонентов, представленных в исходном коде, двоичных или выполняемых файлах. Связи в данном типе диаграммы представляют зависимости одного компонента от другого и имеют специальное отображение через значок «зависимости».

Также данный тип диаграммы позволяет получить представление о поведении компонентов по предоставляемому ими интерфейсу. Интерфейс показывает взаимодействие компонентов, и хотя значки интерфейса принадлежат логическому представлению системы, они могут присутствовать и на диаграмме компонентов.

В текущей модели может быть создано несколько диаграмм компонентов для отражения контейнеров, компонентов верхнего уровня или описания содержимого каждого контейнера компонентов. В последнем случае диаграмма компонентов принадлежит тому контейнеру, для которого отражено содержимое.

Замечания по созданию диаграммы компонентов

Сейчас будет несколько преждевременно создавать полноценную диаграмму компонентов системы управления тепличным хозяйством, так как еще не определены все связи классов и структура наследования.

Однако для небольшой системы, каковой является тепличное хозяйство, состоящей из одного выполняемого файла, разработка такой диаграммы вообще не является целесообразной. Но для того, чтобы ознакомиться с возможностями диаграммы компонентов и не отвлекаться на нее в дальнейшем, предлагается завершить предварительное проектирование системы именно этой диаграммой.

Только предварительное, потому что жизнь сложнее всяких правил. Существуют стратегические и тактические решения, которые иногда противоречат друг другу.

На всем протяжении проектирования системы, вплоть до выхода готового программного продукта, в диаграммы будут вноситься изменения. Мы не будем полностью отражать этот итерационный процесс, поэтому некоторые принятые во время проектирования системы решения могут не совпадать с полученным в конце кодом. Это нормально. Во время разработки необходимо просто отразить эти изменения на созданных ранее диаграммах, что вы можете сделать самостоятельно. А пока создадим диаграмму компонентов лишь для нескольких классов, чтобы получить практику работы с данным типом диаграмм.

В Rational Rose заложена прекрасная возможность работы с программными библиотеками. Причем можно как разрабатывать библиотеки, так и пользоваться уже готовыми. Для этого необходимо лишь указать, какие классы в каких компонентах будут находиться.

При разработке программного проекта разделение классов по компонентам является серьезной задачей. Для того чтобы обеспечить минимальные трудозатраты на разработку и сопровождение, тесно связанные между собой классы собираются в библиотеки DLL, OCX и т.п. Этим обычно занимаются системные аналитики, которые проектируют структуру программного обеспечения, Rational Rose предоставляет все возможности для такого проектирования.

Создание диаграммы компонентов

Для создания диаграммы используйте меню Browse=>Component Diagram или воспользоваться значком на панели инструментов.

При этом будет активизировано диалоговое окно выбора диаграммы, посредством которого пользователь может создавать, удалять, переименовывать диаграммы.

Строка инструментов

При активизации диаграммы строка инструментов приобретает следующий вид (рис.59).

Рис.59. Строка инструментов Component диаграммы

Component (компонент)

Значок Component позволяет показать создание компонента. Компонент представляет собой модуль программного обеспечения, такой как исходный код, двоичный файл, выполняемый файл, динамически подключаемые библиотеки и т.д.

Взаимодействие элементов представляется на диаграмме одним или несколькими значками интерфейса.

Компоненты также могут использоваться для показа взаимосвязи модулей на этапе компиляции или выполнения программы, а также показывать, какие классы используются для создания определенных компонентов.

В связи с тем, что система может состоять из модулей различного типа, пользователь может использовать стереотипы для определения этих различий, причем изменение стереотипа часто ведет к изменению графического отображения компонента на диаграмме.

Обычно имя компонента совпадает с именем файла, представляющего компонент и для каждого компонента должен быть назначен язык программирования.

Package (контейнер)

Значок Package позволяет отобразить контейнер, который объединяет группу компонентов в модели.

Dependence (зависимости)

Пользователь может установить связи данного типа между компонентами, которые изображаются как пунктирная стрелка от одного компонента к другому. Этот тип связей показывает, что классы, содержащиеся в контейнере-клиенте наследуются, содержат элементы, используют или каким-либо другим образом зависят от классов, которые экспортируются из компонента-сервера.

Пользователь может отобразить связь компонента и интерфейса другого компонента и это означает, что компонент-клиент использует операции другого компонента (рис. 60). Если в этом случае связь будет направлена в другую сторону, то это означает, что компонент-клиент предоставляет операции компоненту-серверу.

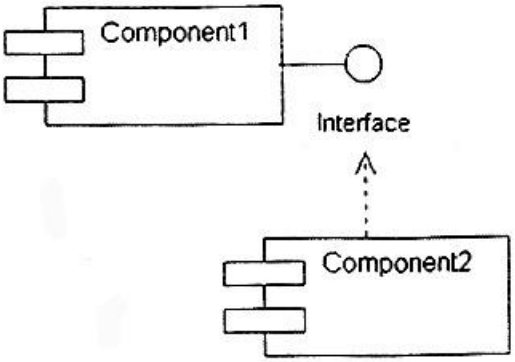

Рис.60. Пример использования операций компонента

Main program (главная программа)
Значок Main program позволяет добавить в модель компонент, обозначающий главную программу. для задач, которые создаются в объектно-ориентированной среде Visual Studio, главная программа скрыта библиотекой MFC, а приложение начинает работать с создания главного класса приложения theApp, который наследуется из библиотечного класса CWinApp, поэтому данный значок используется редко.
Subprogram body (тело подпрограммы)

Значок Subprogram body позволяет добавить в модель компонент, обозначающий тело подпрограммы, и используется также для необъектно-ориентированных компонентов.
Package specification/body (определение/тело контейнера)

Значки Package specification/body позволяют отобразить определения контейнера (Package specification) и описание контейнера (Package body), которые обычно связаны между собой.

Для языка C++ Package specification— это заголовочный файл с расширением .h, а Package body— это файл с расширением *.cpp.
Task specification/body (определение/тело задачи)

Значки Task specification/body позволяют отобразить независимые потоки в многопотоковой системе. Если такие задачи компилируются независимо от обычных модулей, то появляется возможность разместить классы и их определения в таких задачах.

Свойства компонента

Для описанного ранее класса EnvironmentalController создадим значок Package specification. Этим мы покажем, что определение данного класса будет находиться в файле EnvironmentalController.h.
Вкладка General (главная)

После создания элемента перейдите во вкладку General его спецификаций при помощи RClick=>Specification=>General и заполните ее, как показано на рис.61.

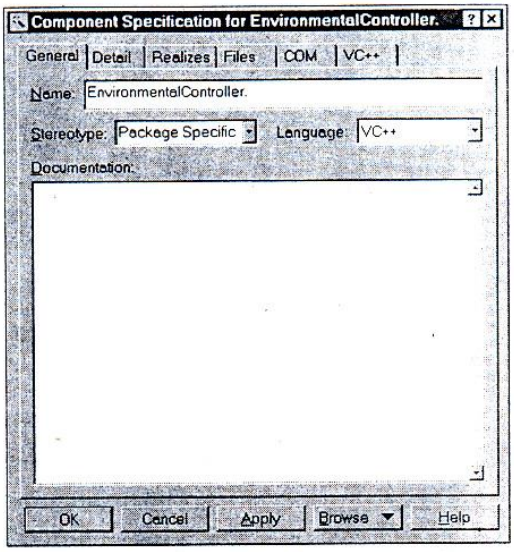

Рис.61. Вкладка General спецификаций компонента

Здесь пользователю доступно изменение имени компонента (Name), дан список стереотипов (Stereotype), доступных для выбора, как и в других элементах, есть возможность заполнения документации (Documentation). Также предоставляется возможность выбора языка программирования для компонента.

В связи с тем, что мы создаем систему управления тепличным хозяйством на Visual C++, необходимо установить этот язык, изменив предлагаемое по умолчанию значение

Analysis, на значение VC++. После чего при следующей активизации данного окна появятся вкладки COM и VC++.

Вкладка COM

Вкладка COM предназначена для установки свойств COM-объектов (рис.62).

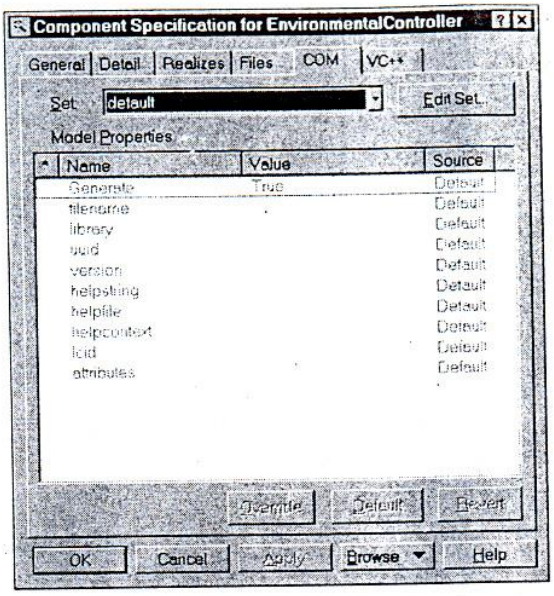

Рис. 62. Вкладка COM спецификаций компонента

На ней отражается список свойств объекта, которые можно изменить. Коротко перечислим эти свойства:

- Generate устанавливает необходимость генерации;
- filename устанавливает полный путь к файлу компонента;
- library устанавливает имя библиотеки, например, stdole;
- uuid устанавливает идентификатор для COM объекта;
- version устанавливает версию компонента;
- helpstring устанавливает строку, которая используется для описания компонента;
- helpfile устанавливает имя файла, в котором содержится справочная система по компоненту;
- helpcontext устанавливает ID темы справки, описывающей данный компонент и находящейся в файле, установленном как helpfile;
- lcid устанавливает локальный идентификатор;
- attributes устанавливает атрибуты, которые в дальнейшем определяют COM компонент или класс, например, control, hidden, restricted, licensed, appobject, nonextensible или oleautomation.

Вкладка VC++

Вкладка VC++ предназначена для отображения свойств объекта, ассоциированного с языком VC++, однако, согласно встроенной документации, эти свойства не предназначены для установки пользователем.

Вкладка Detail

Вкладка Detail (детализация) показывает описание определений для компонента, таких как имя класса, переменные и другие конструкция, зависящие от конкретной реализации языка программирования.

Вкладка Realizes

Вкладка Realizes спецификаций компонента позволяет показать включаемые в компонент классы, а также включить или исключить такие классы из компонента, при этом классы, включенные в компонент, отмечены значком (рис. 63).

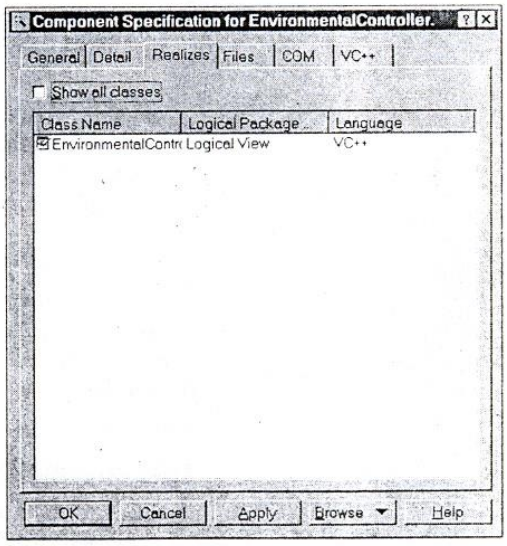

Рис. 63 Вкладка Realizes спецификаций компонента

Флажок Show all classes (показать все классы) позволяет показать только включенные в данный компонент классы или все классы, имеющиеся а модели, например, для дальнейшего включения их в компонент.

Вкладка Files

На вкладке Files представлен список файлов и URL, которые присоединены или добавлены а компонент.

Данная возможность используется для управления связями с дополнительными документами, которые указывают сведения по генерации системы или компонента. Из данного окна возможно открытие и просмотр связанных документов путем двойного нажатия мышкой на строке документа или из контекстного меню.

Создание диаграммы

Теперь создадим отображение остальных компонентов, связанных с контроллером. Создадим значки сенсоров и устройств. Соединим значки связями Dependency таким образом, чтобы показать, что для заголовочного файла контроллера требуются заголовочные файлы устройств и сенсоров, которые, в свою очередь, используются для компиляции самих файлов сенсоров и устройств так, как это показано на рис. 64.

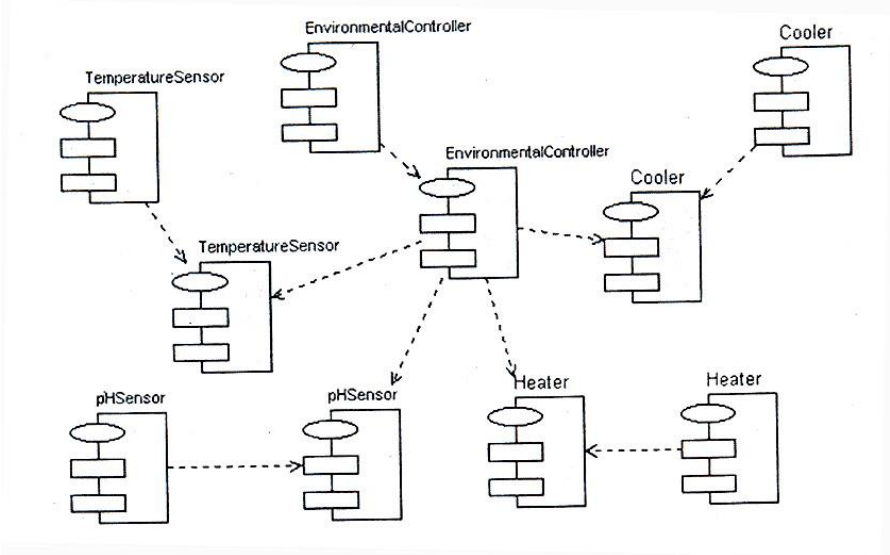

Рис. 64. Диаграмма компонентов