Занятие 12

Rhapsody

В ходе этой лабораторной работы мы установим *IBM Rational Rhapsody Developer*, дополнительный адаптер операционной системы, сгенерируем из UML-модели приложение для QNX Neutrino и запустим его в режиме анимации.

12.1. Получение дистрибутива

Существует два варианта использования Rhapsody — в виде отдельного приложения и в виде плагина Eclipse. Оба варианта содержатся в одном дистрибутиве. Чтобы найти страничку для скачивания дистрибутива Rhapsody на сайте IBM выполните следующие шаги:

1. Откройте в браузере веб-страничку www.ibm.com.
2. В меню выберите элементы Support & downloads | Download | Trials and Demos. Откроется страничка, на которой размещены сгруппированные по алфавиту ссылки на продукты IBM.
3. В секции R перейдите по ссылке Rational Rhapsody, затем выберите интересующий вас продукт (Rhapsody Developer на момент написания книги был доступен по ссылке "Rational Rhapsody Evaluation V7.5.0 Multiplatform"). Откроется страничка скачивания продукта (рис. 12.1). Для скачивания необходимо пройти регистрацию на сайте IBM.

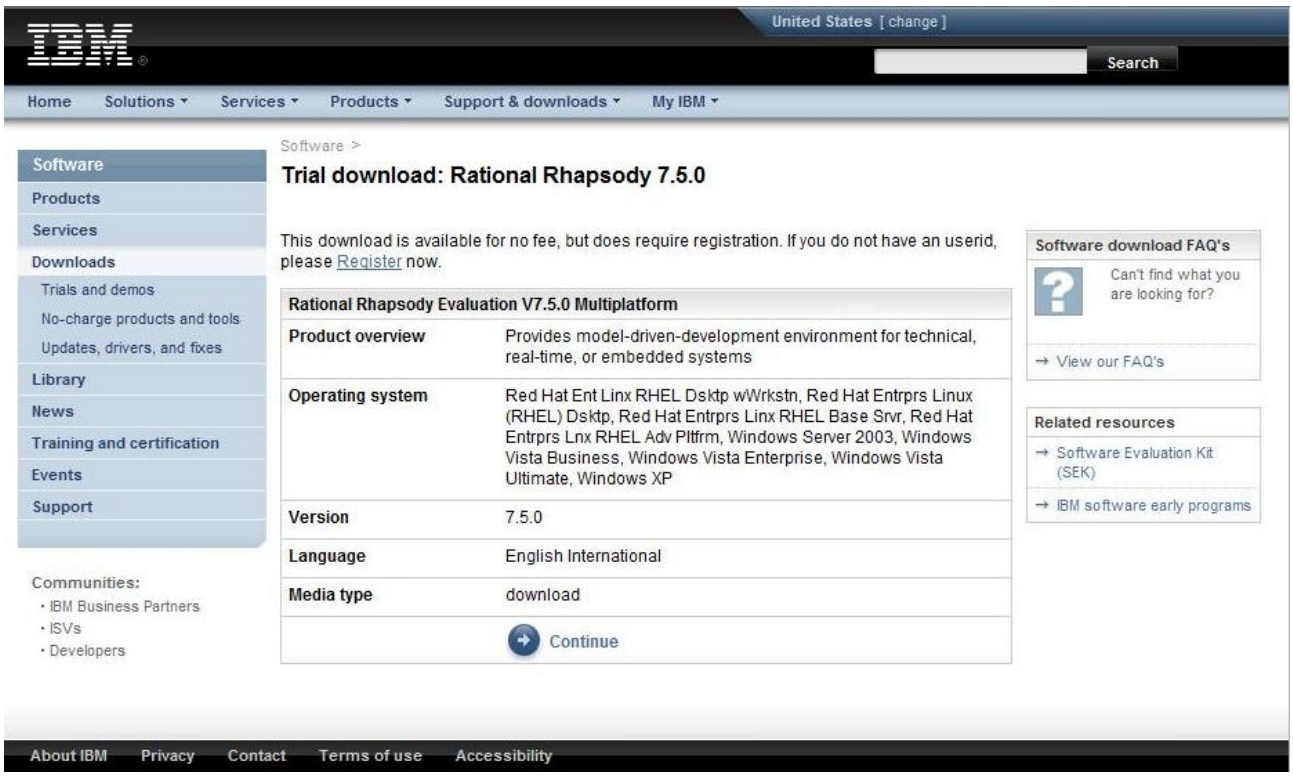

Рис. 12.1. Страница скачивания оценочной версии Rhapsody на сайте IBM

4. Если вы зарегистрированы (если нет — это можно сделать прямо сейчас перейдя по ссылке Register), то можно нажать кнопку Continue для начала процедуры скачивания. Во время процедуры вас попросят заполнить небольшой опросник, затем предложат скачать

дистрибутив для инструментальной системы Windows и/или Linux, а так же файл лицензионного ключа (Rhapsody_Eval_Key.dat).

Для этой лабораторной работы нам потребуется адаптер для QNX Neutrino 6.4. Надо сказать, что на момент написания этого материала текущий дистрибутив Rhapsody Developer V7.5 содержит адаптер для QNX Neutrino 6.3, поэтому более свежий адаптер следует запросить у компании SWD Software (www.swd.ru) — партнера IBM в России по продуктовой линейке Rational Rhapsody. Впрочем, при выходе очередного дистрибутива Rhapsody состав входящих в него адаптеров обновляется.

12.2. Установка Rhapsody Developer и настройка генерации кода для QNX Neutrino

Процедура установки Rhapsody вполне обычна для соответствующей хост-платформы. При использовании платформы Windows необходимо дважды щелкнуть мышью на пиктограмме файла ratlRhapsody_7-5-0_Windows.msi в файловом менеджере и следовать инструкциям программы-инсталлятора.

Учтите, что дистрибутив включает все четыре продукта Rhapsody, упоминаемых на пятом занятии (два для системных инженеров и два для инженеров по разработке программного обеспечения), а так же все пакеты расширения. Для данной лабораторной работы необходимо выбрать установку IBM Rational Rhapsody Developer (рис. 12.2).

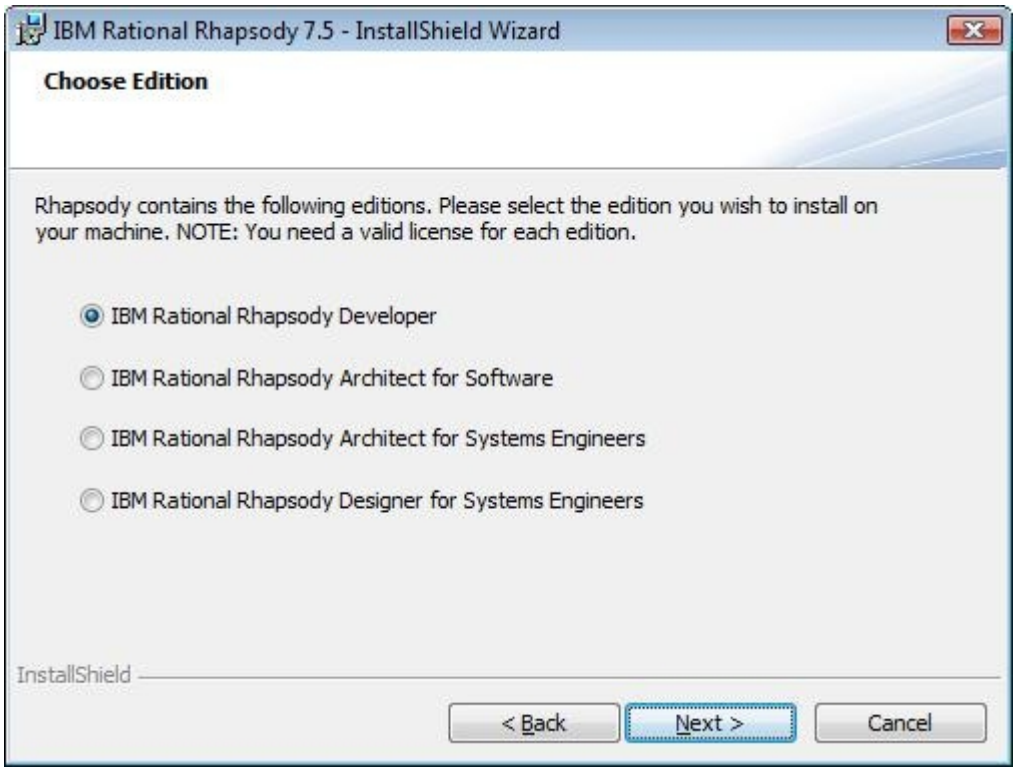

Рис. 12.2. Выбор продукта из линейки Rhapsody для инсталляции

Вам будет предложено указать каталог для установки продукта (у меня D:\Rhapsody7.5). Rhapsody Developer поддерживает генерацию кода для языков C, C++, Java и Ada — инсталлятор предложит выбрать любую их комбинацию. Чтобы устанавливать на диск поменьше компонентов, я выбрал только язык C++ (рис. 12.3).

Рис. 12.3. Выбор языка и адаптера для генерации исходных текстов.

Здесь же с помощью флажка Check for Real Time OS Settings можно "попросить" инсталлятор включить в процедуру выбора адаптеров операционных систем из входящих в поставку (рис. 12.4).

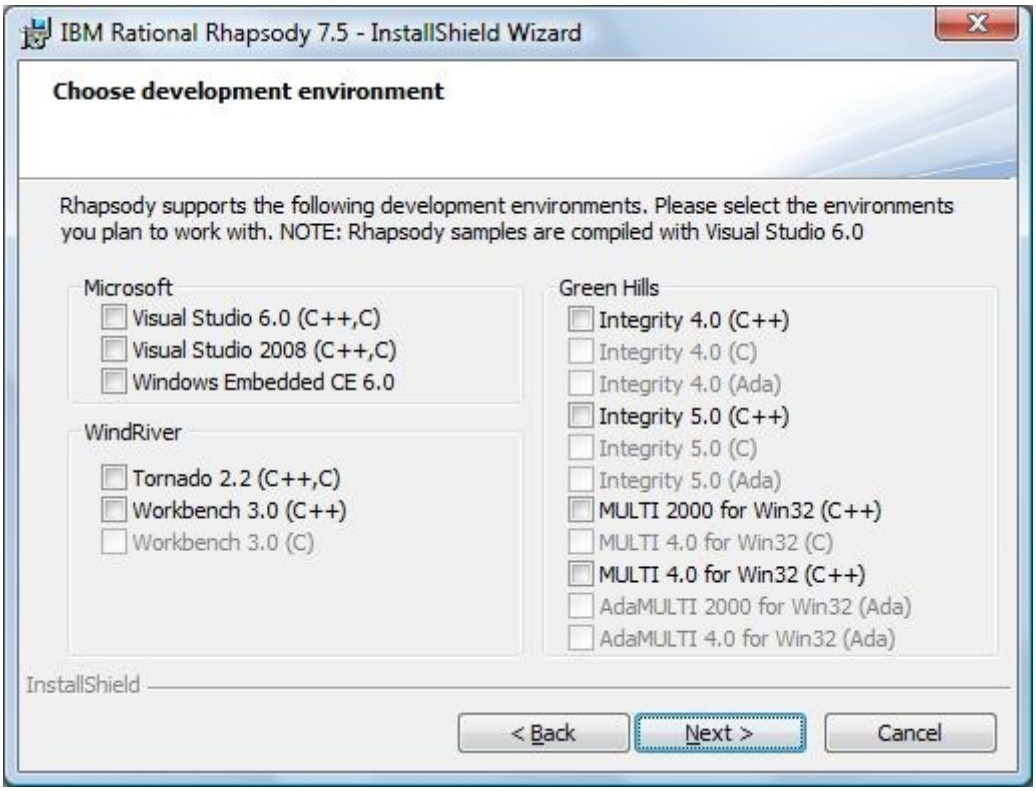

Рис. 12.4. Выбор первой "порции" адаптеров для генерации исходных текстов

Если нажать кнопку Next, то нам будет предложена следующая "порция" адаптеров (рис. 12.5).

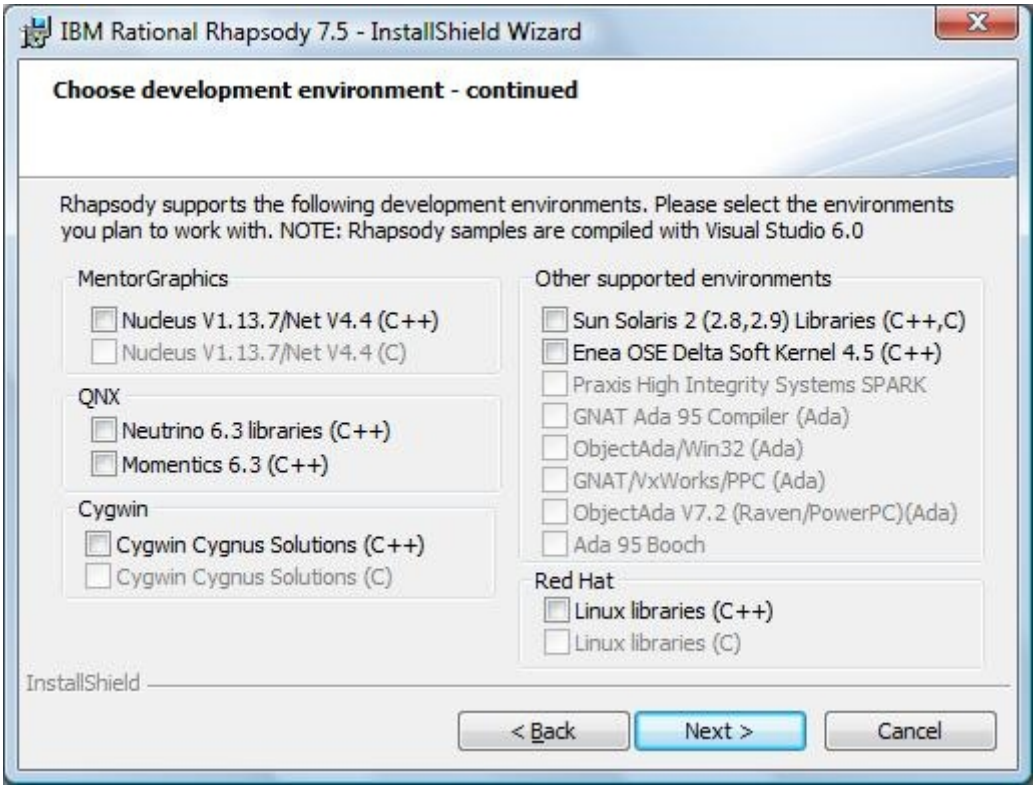

Рис. 12.5. Выбор второй "порции" адаптеров для генерации исходных текстов

На момент написания этого материала дистрибутив Rhapsody Developer содержит адаптер для QNX Neutrino 6.3. Поэтому я не отметил ни одного адаптера — после установки Rhapsody мы будем устанавливать дополнительный адаптер QNX Neutrino 6.4. Это полезная процедура, т. к. некоторые адаптеры созданы независимыми производителями и в дистрибутив не входят (например, адаптер QNX 4.25).

После очередного нажатия кнопки Next для выбранных адаптеров будет предложено указать каталоги размещения инструментальных средств (рис. 12.6).

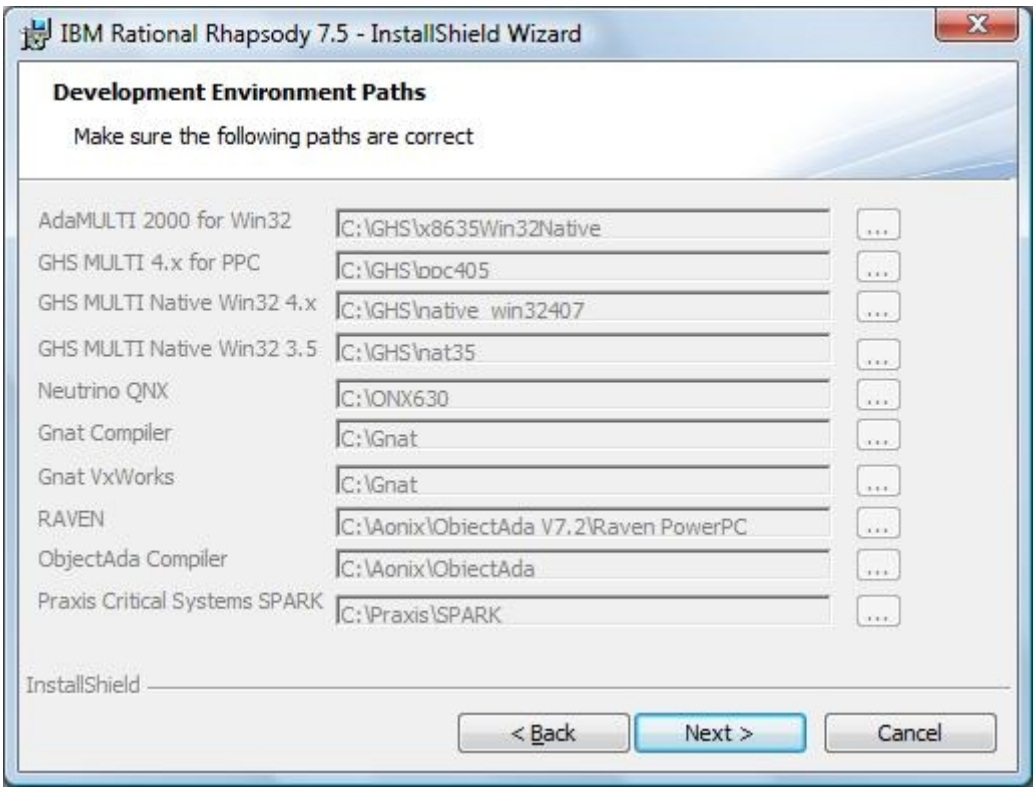

Рис. 12.6. Настройка путей размещения инструментальных средств

Затем инсталлятор предложит выбрать, какие из штатных расширений Rhapsody необходимо установить. Для данного занятия расширения не требуются, поэтому я двигаюсь дальше, не выбирая ничего (рис. 12.7).

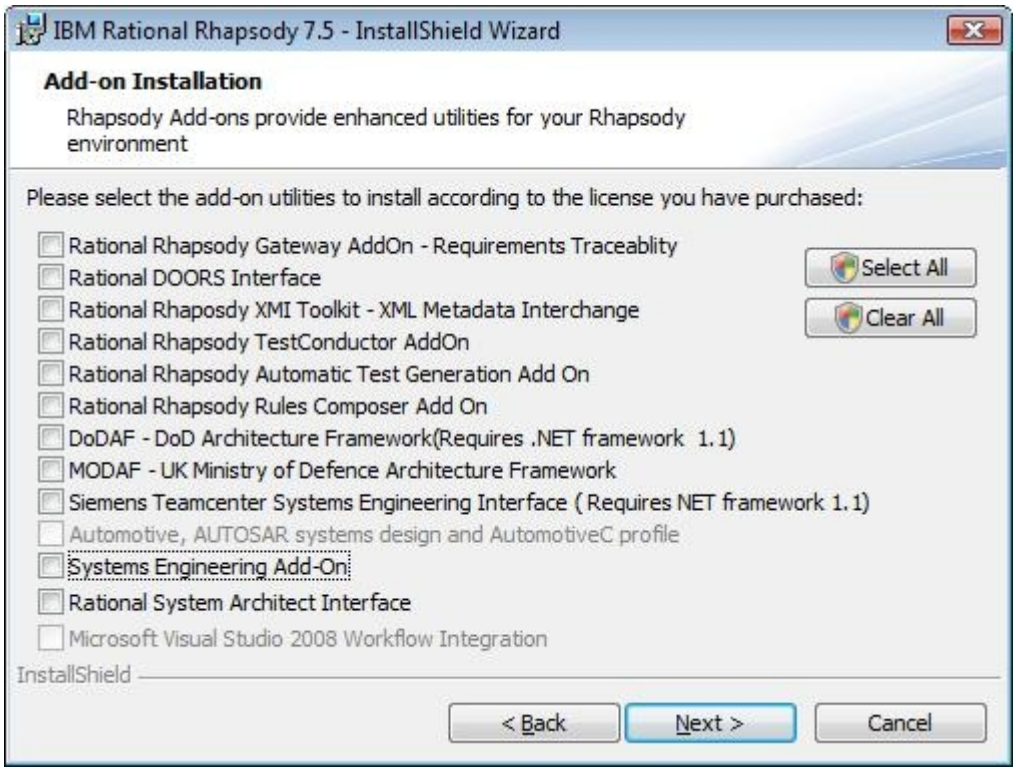

Рис. 12.7. Выбор инсталляции продукта-расширения для Rhapsody

В качестве одного из заключительных шагов потребуется указать скаченный с сайта IBM

файл лицензионного ключа (рис. 12.8).

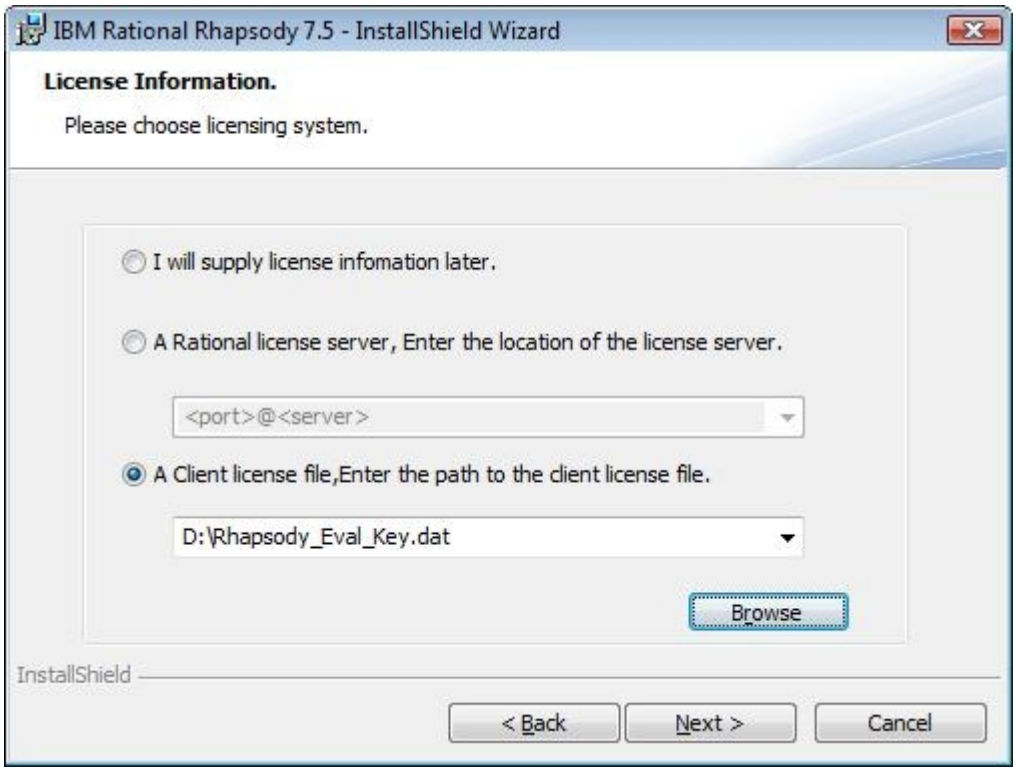

Рис. 12.8. Предоставление лицензионной информации

Теперь Rhapsody Developer установлен, но необходимо добавить адаптер QNX Neutrino 6.4. Адаптер поставляется в виде архива Rhapsody75_QNX640_Adapter.zip. В архиве есть папка Share, содержимое которой следует скопировать в одноименную папку инсталлированного дистрибутива Rhapsody. При этом необходимо перезаписать все файлы с одинаковыми именами. Затем в папке Share\Properties надо переименовать файл sample_siteC++.prp в siteC++.prp. Содержимое файла такое:

```
Include "siteC++_OXF_QNX640_x86.prp"
Subject CPP_CG
        Metaclass QNX640_x86
                Property RemoteHost String "192.168.168.3"
    end
        Metaclass Configuration
                Property Environment Enum "MinGW,QNX640_x86" "MinGW"
        end

end
end
```

MinGW в данной лабораторной работе не требуется. Хотя в принципе, это нужный продукт, который может использоваться для генерации приложений Windows. IP-адрес

"192.168.168.3" используется для анимации и его следует заменить на реальный IP-адрес вашей инструментальной системы. Заменять есть смысл тогда, когда вы используете статический IP-адрес — мой компьютер получает параметры IP по протоколу DHCP, поэтому мне незачем модифицировать запись. Текущий IP-адрес своей хост-системы мне придется задавать в свойствах проекта — я чуть позже покажу вам, как это делается.

И, наконец, необходимо в папке Share\etc модифицировать файл qnx640make.bat так, что бы он задавал корректные значения переменных QNX_HOST и QNX_TARGET (по умолчанию заданы, соответственно, C:/QNX640/host/win32/x86 и C:/QNX640/target/qnx6).

Rhapsody Developer установлен и готов к разработке приложений для QNX Neutrino 6.4.1.

12.3. Пример "Dishwasher"

Эта книга не является учебником по Real-Time UML (не спорю, может быть и зря ☺). Поэтому мы, не долго думая, возьмем проект из уже готовых примеров, поставляемых в составе среды разработки Rhapsody.

Для этого запустим Rhapsody и выберем элемент File | Open. Затем с помощью диалогового окна открытия файлов найдем в установочном каталоге Rhapsody (у меня это D:\Rhapsody7.5) файл Samples\CppSamples\Dishwaser\Dishwasher.rpy. Откроется проект-пример.

Как и все файлы конфигурации Rhapsody-проекта и создаваемой в нем UML-модели, файл Dishwasher.rpy написан на языке XML. Таким образом, модель представляет собой, по сути дела, набор текстовых файлов, что очень удобно с точки зрения систем управления версиями и конфигурационного управления. Кстати, чтобы был более понятен смысл модели замечу, что "dishwasher" по-английски означает "посудомоечная машина".

В левой части рабочего пространства Rhapsody размещается навигатор модели, с помощь которого можно получить доступ к любым артефактам модели (рис. 12.9).

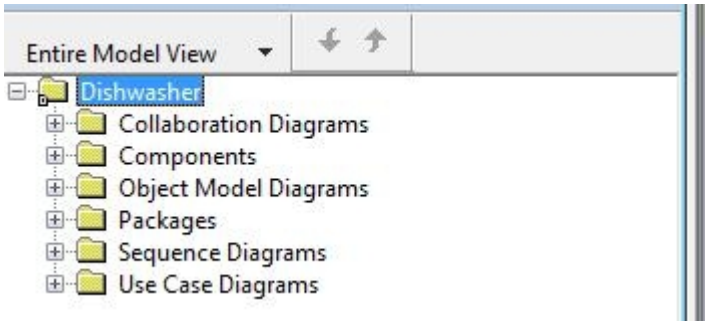

Рис. 12.9. Навигатор модели Rhapsody

Навигатор включает несколько логических папок, облегчающих поиск соответствующих артефактов модели. Логические папки расположены по алфавитному порядку, при этом их можно разделить на три категории:

- Папка Packages (Пакеты). Идея (или, в терминах UML, *семантика*) пакета в языке UML очень похожа на идею, т. е. семантику, пакета в языке Java. Таким образом, пакет является, во-первых, способом логической группировки артефактов UML-модели, во-вторых, предоставляет независимое *пространство имен* (имя каждого элемента UML-модели должно быть уникальным только в пределах пакета). Кроме того, содержимое некоторых пакетов может выполнять для UML-модели ту же роль, что библиотеки функций для программ на языке C.

- Папки диаграмм (Collaboration Diagrams, Use Case Diagrams и т. д.). В эти логические папки "помещаются" почти все создаваемые диаграммы в соответствии с их типами. "Почти" означает, что диаграммы деятельности и конечных автоматов размещаются несколько иначе — об этом мы еще поговорим далее.
- Папка Components. В этой логической папке "размещаются" создаваемые программные компоненты — библиотеки и исполняемые файлы.

Раскроем все логические папки и посмотрим, что в них лежит (рис. 12.10).

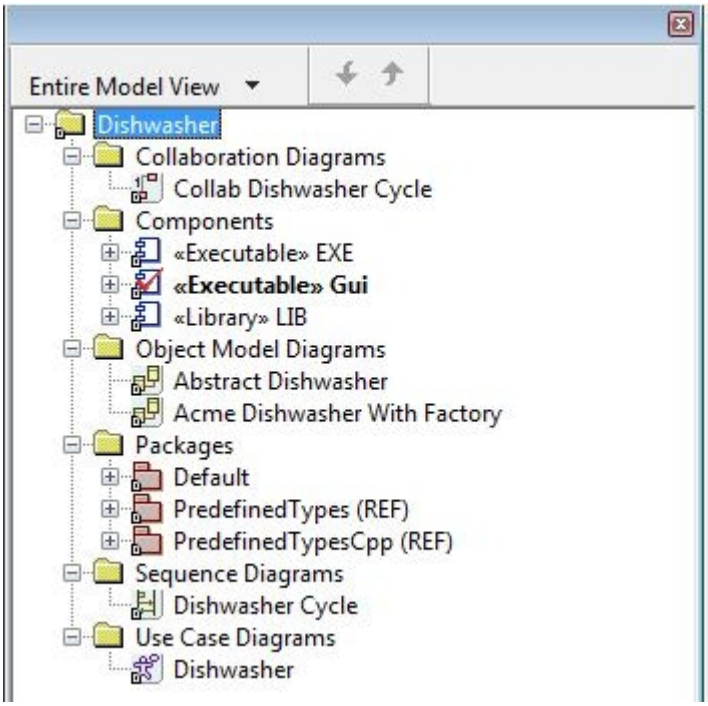

Рис. 12.10. Содержимое модели

На рис. 12.10 мы видим:

- пять различных диаграмм, из которых состоит модель. Двойной щелчок мышью на имени диаграммы приводит к ее открытию в правой части окна Rhapsody. Откроем, например, диаграмму классов AbstractDishwasher (рис. 12.11);
- три компонента (EXE, Gui, LIB). Компонент EXE — это собственно программа, генерируемая из модели. Компонент LIB является библиотекой для компонента Gui. Компонент Gui — это Windows-приложение, предоставляющее графический интерфейс пользователя к программе EXE.
- три пакета (Default, PredefinedTypes, PredefinedTypesCpp). Пакеты PredefinedTypes и PredefinedTypesCpp являются хорошими образцами пакетов-библиотек. Они не входят в проект, так сказать, физически, а являются ссылками на внешние пакеты, о чем напоминает слово "REF" в скобках возле имени каждого из пакетов-библиотек. Что касается пакета Default, то в данном проекте он является единственным пакетом, содержащим все артефакты UML-модели. В связи с чем заслуживает чуть больше внимания.

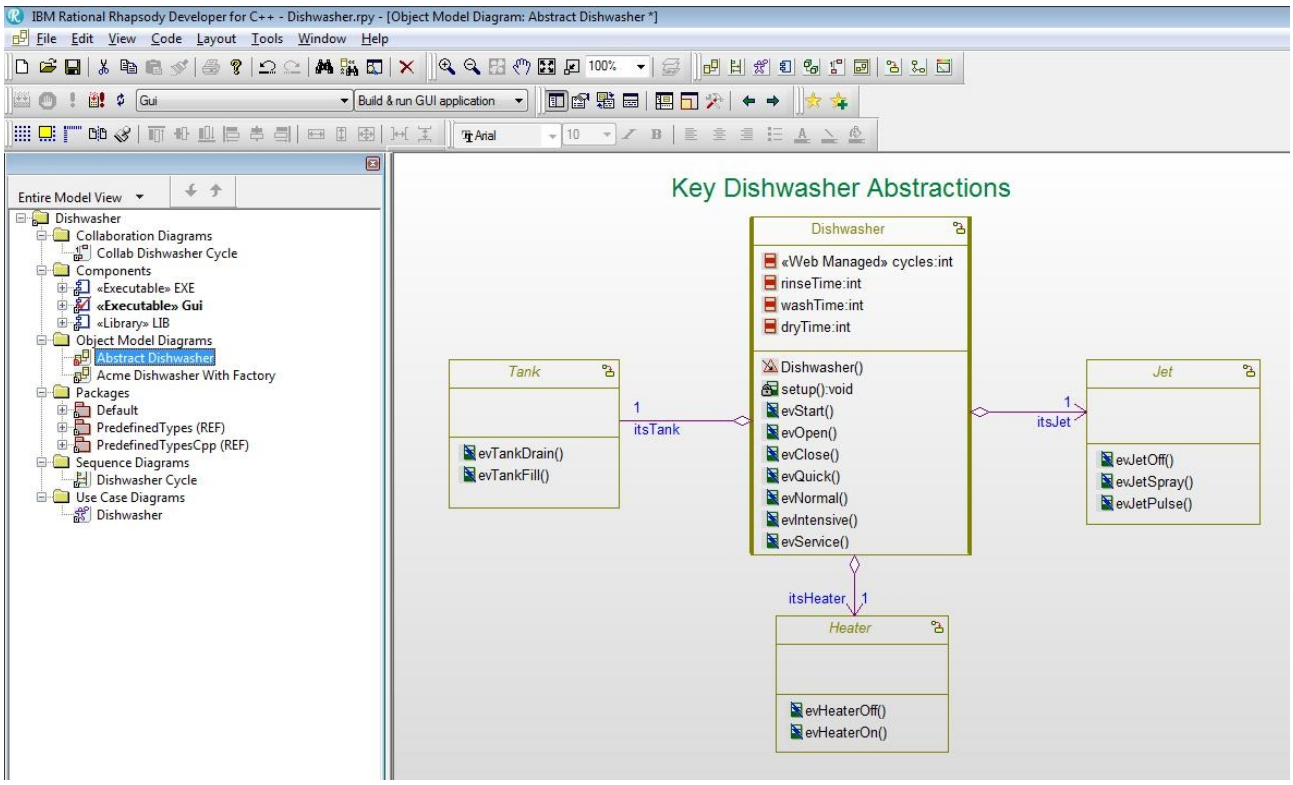

Рис. 12.11. Диаграмма классов AbstractDishwasher

С помощью пакет Default логической папки Packages мы можем получить доступ ко всем UML-элементам модели, за исключением диаграмм. Диаграммы, как вы помните, "хранятся" в своих логических папках. Удаление диаграммы не приводит к автоматическому удалению размещенных на ней элементов модели. Мало того, мы можем размещать один элемент на нескольких диаграммах, "перетаскивая" его мышью из пакета в открытую справа диаграмму.

Откроем пакет Default на два уровня вниз (рис. 12.12).

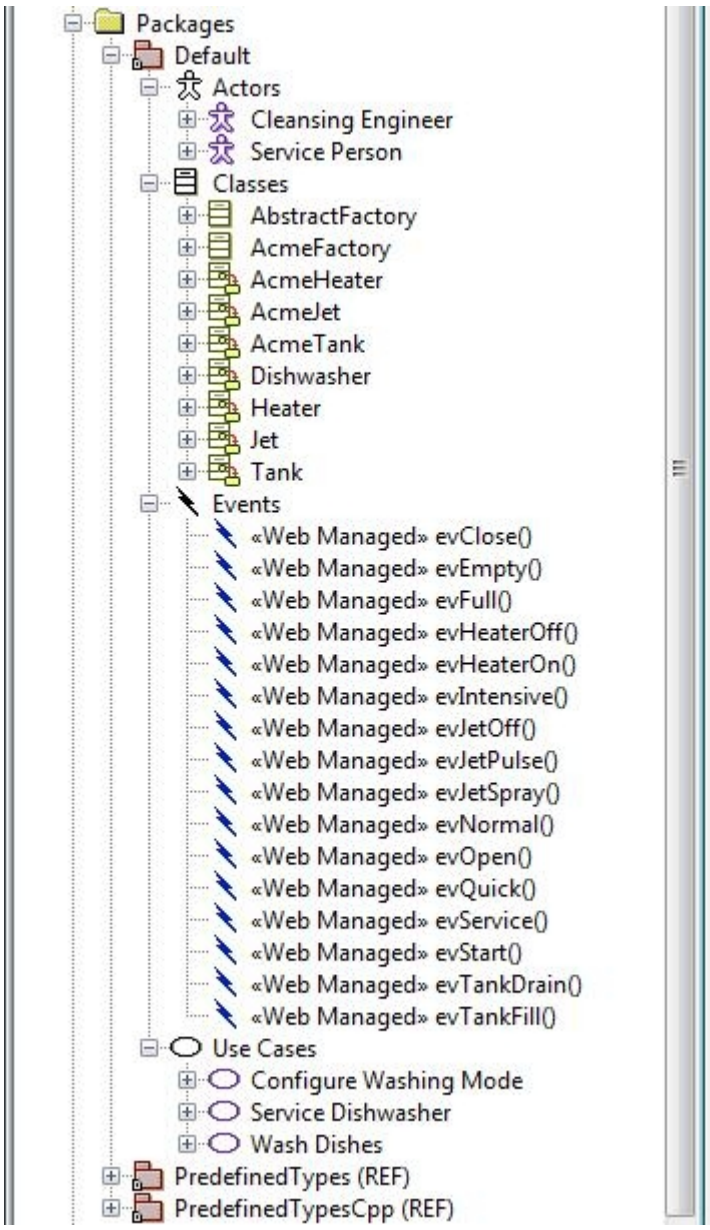

Рис. 12.12. Пакет Default проекта-примера Dishwasher

Мы видим, что пакет содержит такие UML-элементы как классы, варианты использования и т. д. Каждый элемент имеет некоторое содержимое, доступное из навигатора — класс, например, имеет атрибуты и методы.

> *Примечание:*
> Элемент типа Event (Событие) это то, что генерирует внешняя среда и что должна обрабатывать модель.

Самым сложным элементом является класс. Дело в том, что объект класса может иметь поведение, моделируемое с помощью конечного автомата или диаграммы деятельности. Доступ к диаграммам поведения для объектов некоторого класса обеспечивается, если в навигаторе развернуть содержимое класса и дважды щелкнуть на составном элементе, обозначающем соответствующую диаграмму (рис. 12.13).

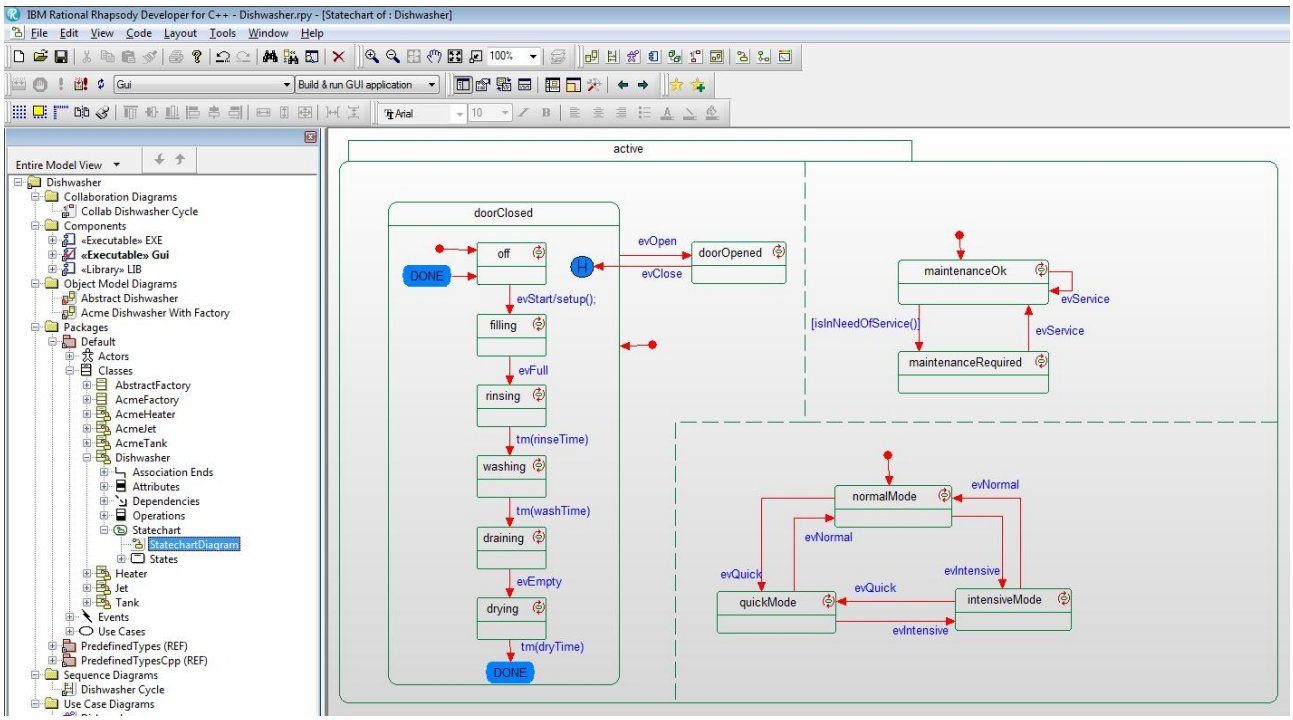

Рис. 12.13. Диаграмма поведения класса Dishwasher

Обратите внимание, что классы, имеющие диаграммы поведения (в терминах UML — *реактивные классы*), обозначаются в пакете Default специальным значком. Этот же значок присутствует в правой верхней части представления реактивного класса на диаграммах (см. рис. 12.11). Если щелкнуть мышью на этом значке на структурной диаграмме, то откроется поведенческая диаграмма для данного класса.

Впрочем, довольно комментариев к примеру Dishwasher. Теперь вы можете самостоятельно исследовать эту модель (как, впрочем, и любой другой пример), открывая и изучая все диаграммы.

12.4. Генерация исполняемого модуля

Приступим к созданию приложения QNX Neutrino из UML-модели Dishwasher. Будущее приложение представлено в навигаторе модели компонентом с именем EXE (ну, как назвали, так назвали).

> *Примечание:*
> Компонент Gui к UML-модели прямого отношения не имеет — он выполняет для компонента EXE функцию, как выражаются системные программисты, "графической "морды"". "Графическая "морда"" (по-правильному GUI или ЧМИ), конечно, зависит от типа графической оболочки конкретной ОС. Компонент Gui иллюстрирует привязку Gui к EXE в среде Windows.

Каждый компонент может иметь произвольное количество так называемых конфигураций. "Конфигурация" задает конкретные параметры генерации модуля, например, для какой операционной системы сгенерировать код, включать ли в двоичный модуль отладочную информацию, инструментировать ли программу для поддержки анимации и т. д.

Давайте же добавим в список конфигураций компонента "EXE" конфигурацию "QNX". Для этого нужно раскрыть в навигаторе элемент модели "EXE", далее на элементе "Configurations" щелкнуть правой кнопкой мыши, выбрать элемент меню "Add New

Configuration" и указать ее имя "QNX".

Чтобы к созданной нами конфигурации применялись все команды генерации кода и компиляции необходимо сделать ее активной. Для этого необходимо щелкнуть правой кнопкой мыши на ее имени в навигаторе и выбрать элемент меню Set as Active Configuration (рис. 12.14).

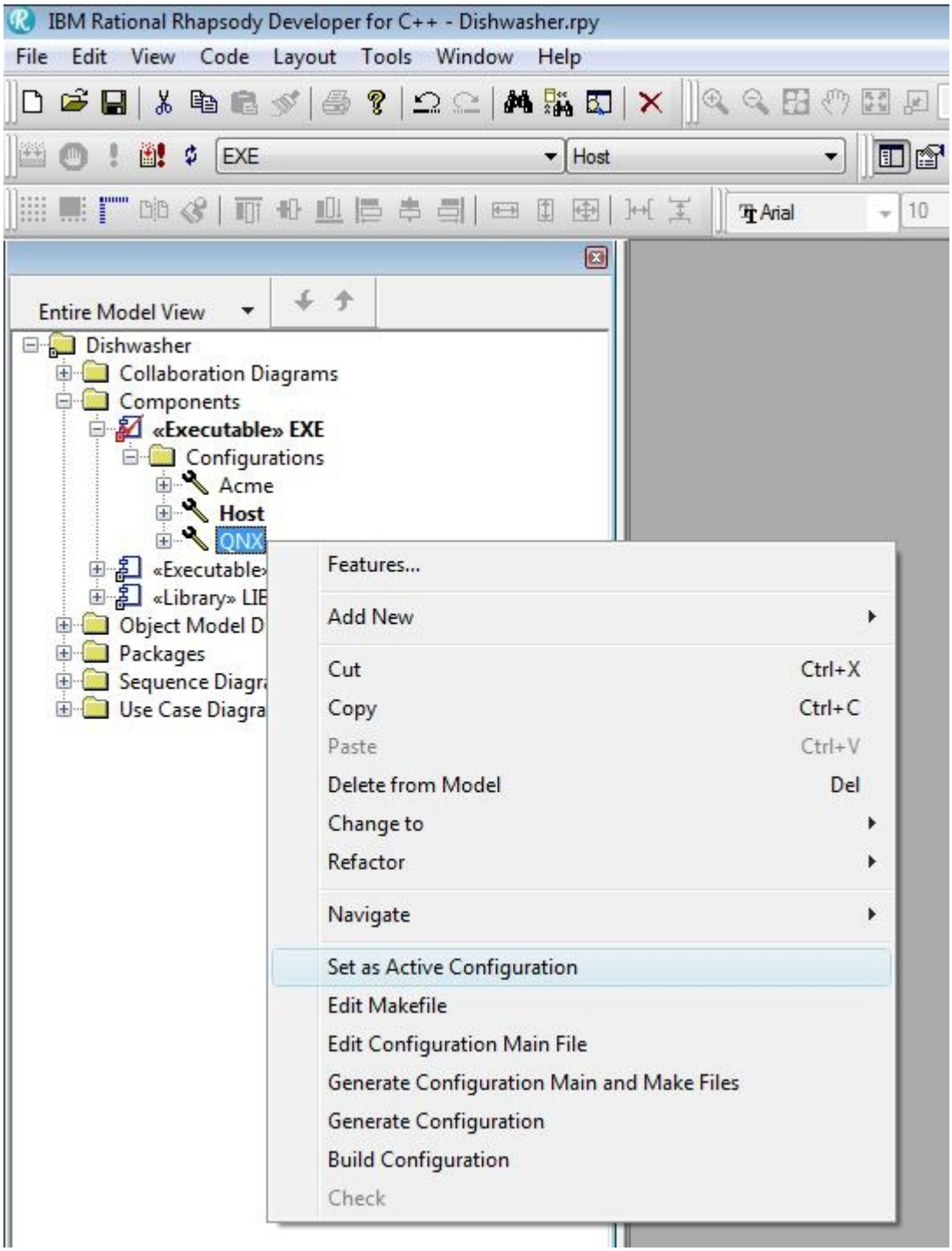

Рис. 12.14. Установка активной конфигурации.

Либо можно воспользоваться панелью инструментов (рис. 12.15). Кстати, панель инструментов позволяет всегда видеть какая конфигурация и для какого компонента в данный момент является активной.

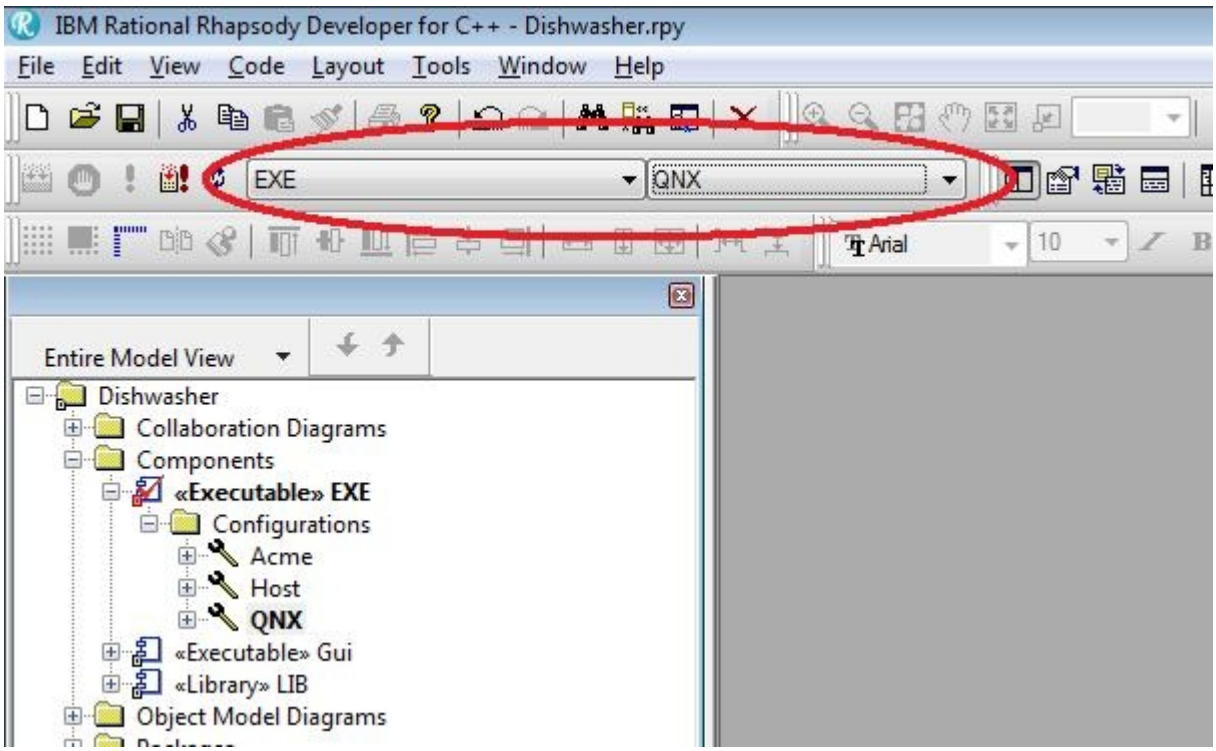

Рис. 12.15. Отображение активной конфигурации

Теперь настроим параметры генерации кода. Для этого необходимо открыть окно настроек для данной конфигурации — для этого необходимо выбрать верхний элемент (Features) того же меню, которое мы использовали чтобы конфигурацию "QNX" активной (рис. 12.14).

В окне настроек нас интересуют три вкладки — Initialization (Инициализация), Settings (Параметры) и Properties (Свойства).

Вкладка Initialization указывает, объекты каких классов необходимо создавать автоматически при запуске приложения. Необходимо указать инициализацию, как показано на рис. 12.16:

- первоначальные реализации классов (Initial instances) должны создаваться явно (Explicit);
- должна быть создана реализация (т. е. объект) класса Dishwasher из пакета Default. Дело в том, что класс Dishwasher является композитным, т. е. *содержит* другие необходимые классы и их ассоциации *внутри* себя. Поэтому при создании экземпляра этого класса экземпляры входящих в него классов будут созданы автоматически и между ними автоматически будут созданы ссылки в соответствии с заданными ассоциативными связями;
- генерировать код для актеров не нужно.

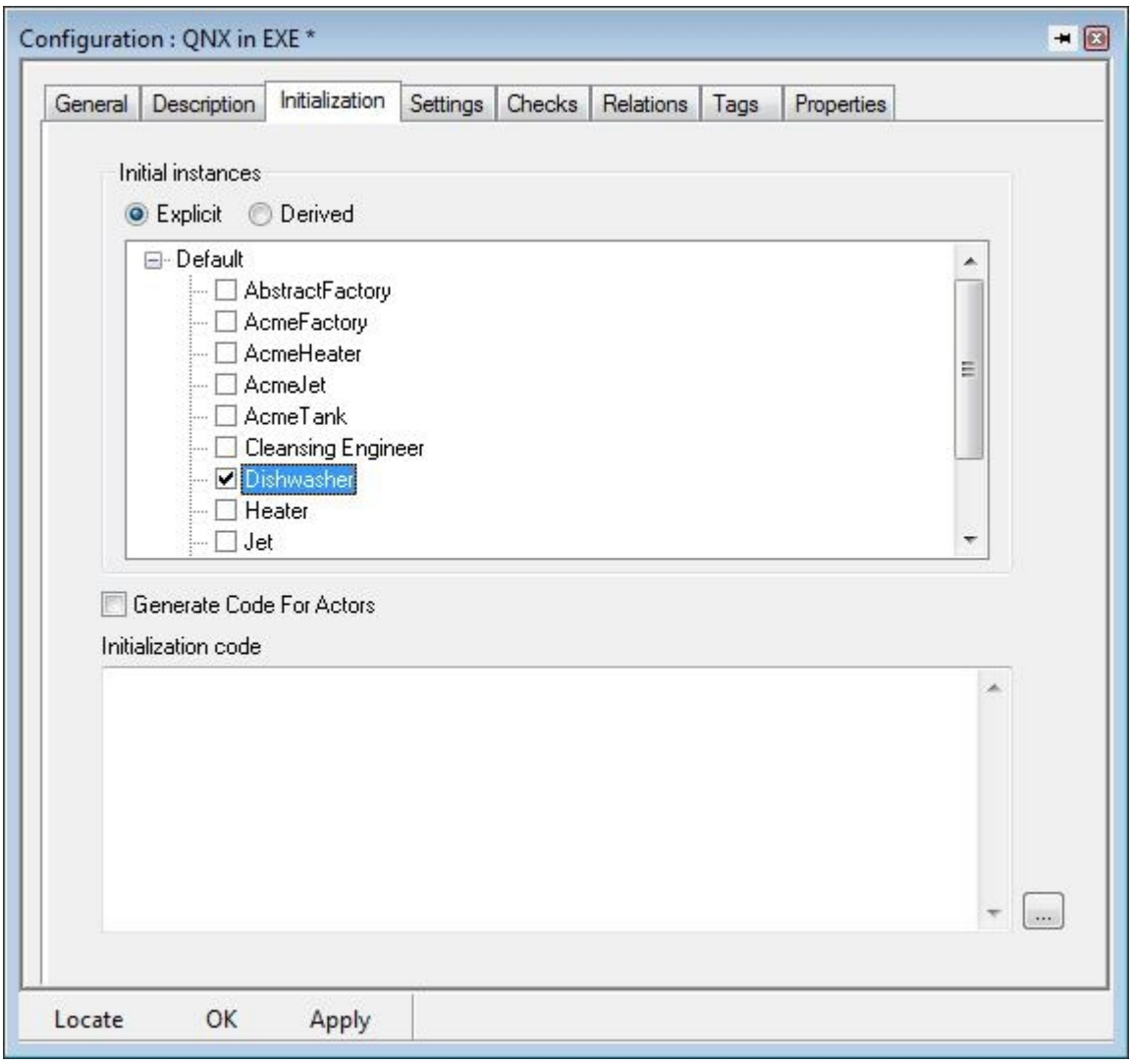

Рис. 12.16. Установки инициализации приложения

На вкладке Settings все оставим по умолчанию, кроме режима инструментирования (Instrumentation Mode) и целевой среды (Environment) — выберем, соответственно, Animation и QNX640_x86 (рис. 12.17).

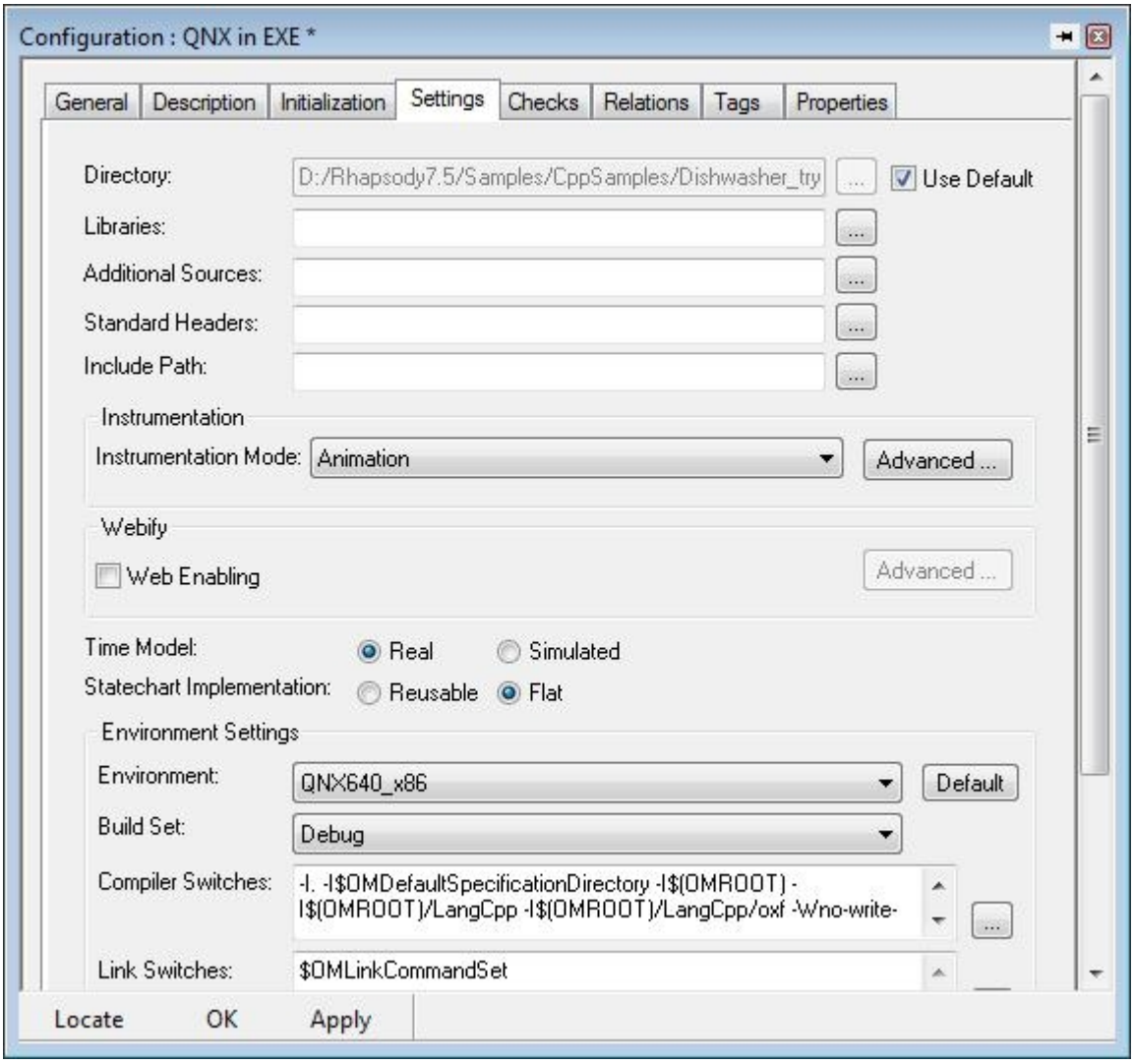

Рис. 12.17. Параметры генерации кода приложения

На рис. 12.17 обратите внимание на такие параметры как Webify, Time Model и Statechart Implementation. Все они достаточно любопытны.

Параметр Webify указывает на то, чтобы в состав исполняемого модуля был включен небольшой веб-сервер с ограниченным функционалом. После запуска такого приложения на целевой системе мы сможем "зайти" на него с помощью веб-браузера и увидеть веб-страничку, которая будет в табличной форме содержать определенные объекты, значения их атрибутов, кнопки запуска методов и посылки им событий. Слово "определенные" означает то, что для отображения на веб-страничке классы и их признаки должны иметь стереотип "Web Managed", как, например, события на рис. 12.12.

По сути дела, эта веб-страничка служит для временной замены GUI пока параллельно с разработкой приложения идет разработка приложения-GUI для него. Наличие такого временного GUI позволяет тестировать, как приложение будет реагировать на сообщения и события, инициируемые GUI.

Параметр Time Model (модель времени) может иметь значение Real или Simulated. В первом варианте приложение будет работать так, как его поведение задано в модели — т. е. если задано ожидание срабатывания таймера через 10 минут, то таймер и на самом деле будет установлен в значение 10 минут. Это не всегда удобно при отладке логики работы приложения — если в некоторых случаях ожидание события на реальном объекте составляет сутки, то ведь это не значит, что тестировщику надо сидеть сутки за компьютером и ждать

наступления события. Для этих случаев предназначен второй вариант, называемый "модельным временем". Модельное время никак не привязано к астрономическому и все таймеры срабатывают без реального ожидания истечения заданного интервала времени.

Параметр Statechart Implementation (группа переключателей) указывает, как в приложении должны быть реализованы конечные автоматы. Выбор переключателя Flat предписывает использовать конструкцию switch-case, где каждый case является каким-либо из состояний. Выбор переключателя Reusable "хитрее" — состояния моделируются отдельными классами. Второй вариант для небольших моделей несколько медленнее да и памяти требует больше, но в сложных приложениях с наследованием поведения объектов может дать выигрыш как с точки зрения скорости, так и с точки зрения требований к объему памяти.

На вкладке Properties (рис. 12.18) нас интересует свойство CPP_CG:QNX640_x86:RemoteHost. Это — IP-адрес хост-системы с Rhapsody, необходимый сгенерированному приложению для доступа к ней при выполнении в режиме анимации. По умолчанию IP-адрес берется из файла конфигурации, который мы обсуждали в рамках *разд. 12.2* этого занятия.

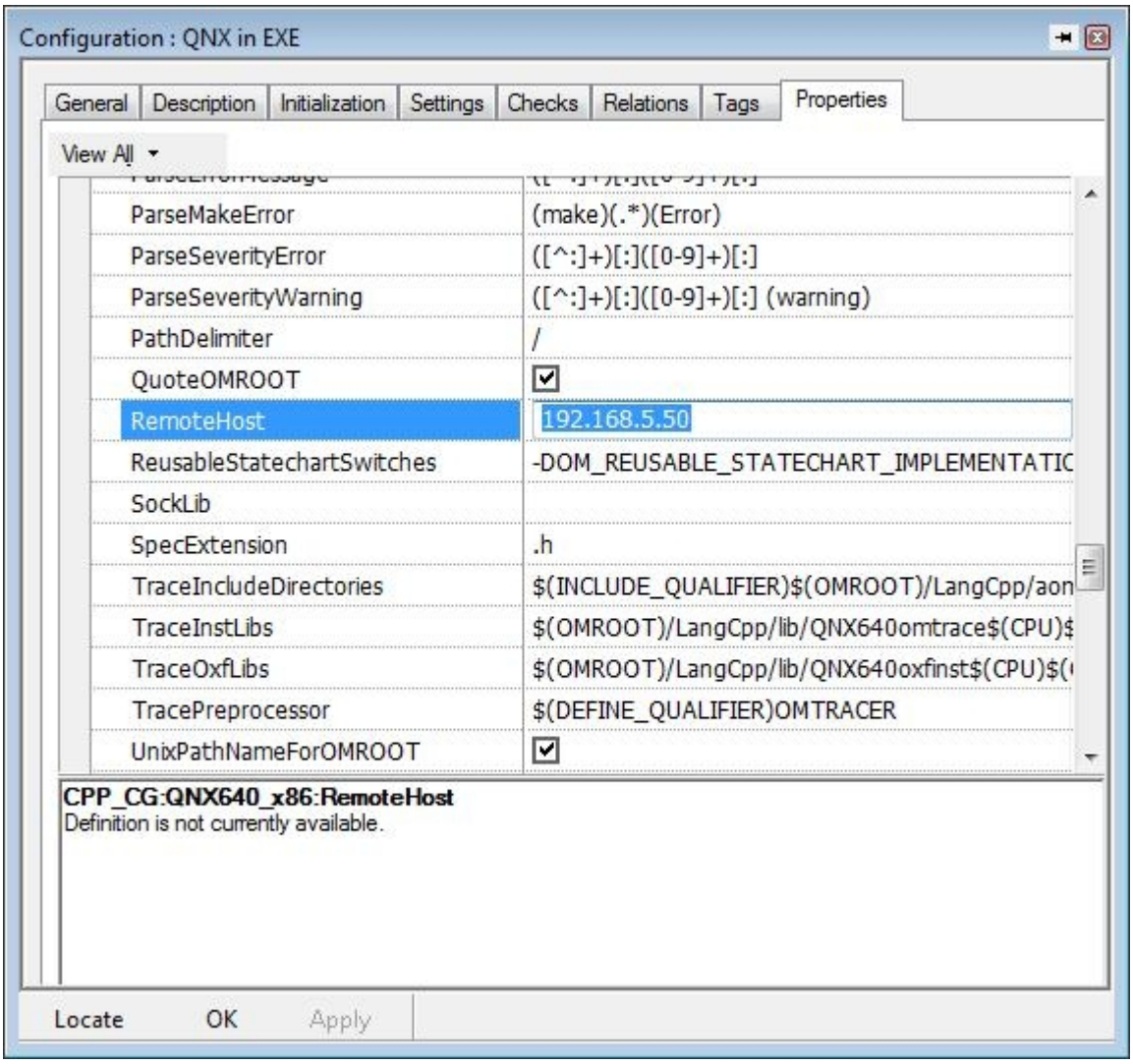

Рис. 12.18. Переопределение свойств проекта Dishwaser

Все сделанные изменения необходимо применить (Apply) и окно настроек конфигурации "QNX" для модуля "EXE" можно закрыть (OK). Среда разработки Rhapsody готова

генерировать программу.

Выберем элемент меню Code | QNX. Нам будет предложено создать каталог для того, чтобы в него класть сгенерированный исходный код, а так же результаты компиляции и компоновки. Соглашаемся. В результате в папке проекта появится каталог EXE\QNX с исходными текстами, сгенерированными из UML-модели.

Теперь выберем элемент меню Code | Build | Buld EXE. В каталоге EXE\QNX появится исполняемый модуль EXE, который можно запускать на целевой системе QNX Neutrino.

Вы, наверное, заметили, что в меню Code есть элемент Run EXE. Чтобы им пользоваться необходимо на целевой системе QNX Neutrino настроить FTP-сервер, а в каталоге Share\etc среды Rhapsody правильно отредактировать сценарий qnx640run.bat. Мы этого не делали, т. к. для копирования приложения на целевую систему и запуска его на ней можно использовать представление "Target File System Navigator" в QNX Momentics IDE.

12.5. Запуск исполняемого модуля в режиме анимации

Итак, скопируйте исполняемый файл EXE на целевую систему QNX Neutrino и запустите его. Поскольку приложение анимировано, оно установит соединение со средой Rhapsody и она перейдет в режим анимации. Вы увидите, что панель управления изменилась (рис. 12.19) и на ней появились элементы управления исполнением приложения.

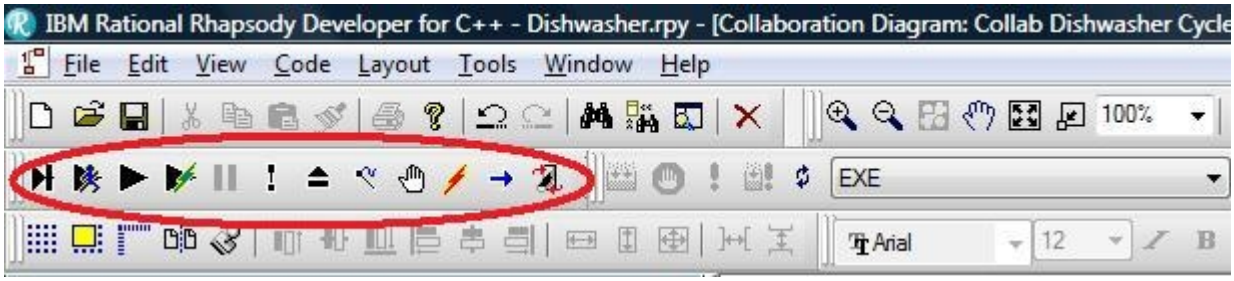

Рис. 12.19. Панель управления Rhapsody в режиме анимации

Кнопки панели управления анимацией аналогичны кнопкам управления символьного отладчика интегрированной среды разработки. Это и не удивительно, так как исполнение UML-моделей в режиме анимации часто называют *динамической отладкой*.

Нажмем вторую кнопку слева. Она называется Go idle и предписывает приложению выполнить инициализацию и остановится. Что значит "выполнить инициализацию"? Это значит, что будет создан экземпляр композитного класса Dishwasher, экземпляры всех входящих в него классов, созданы и проинициализированы указатели объектов друг на друга в случае, если классы объектов имеют соответствующие ассоциативные связи. Конечный автомат каждого из созданных объектов перейдет в начальное состояние.

Созданные объекты отобразятся в навигаторе модели — у соответствующих классов появится вложенный элемент Instances (реализации), содержащий список созданных объектов данного класса (рис. 12.20).

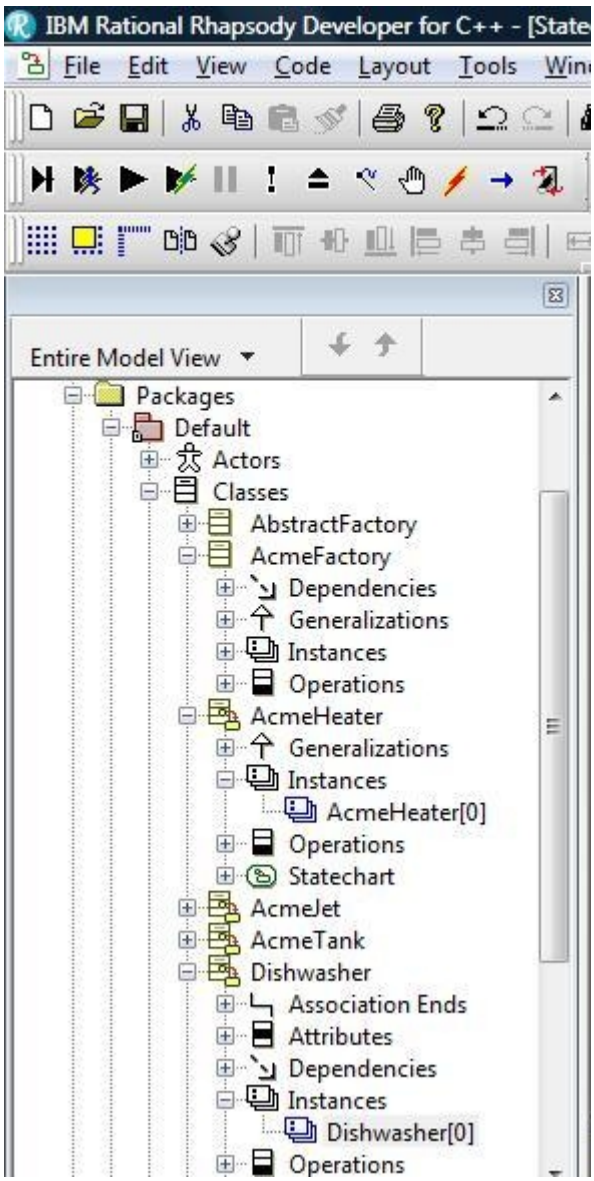

Рис. 12.20. Отображение экземпляров классов в навигаторе модели

Например, класс Dishwasher будет содержать объект Dishwasher[0]. Если на имени объекта щелкнуть правой кнопкой мыши, то можно открыть анимированную диаграмму конечного автомата для данного объекта (рис. 12.21).

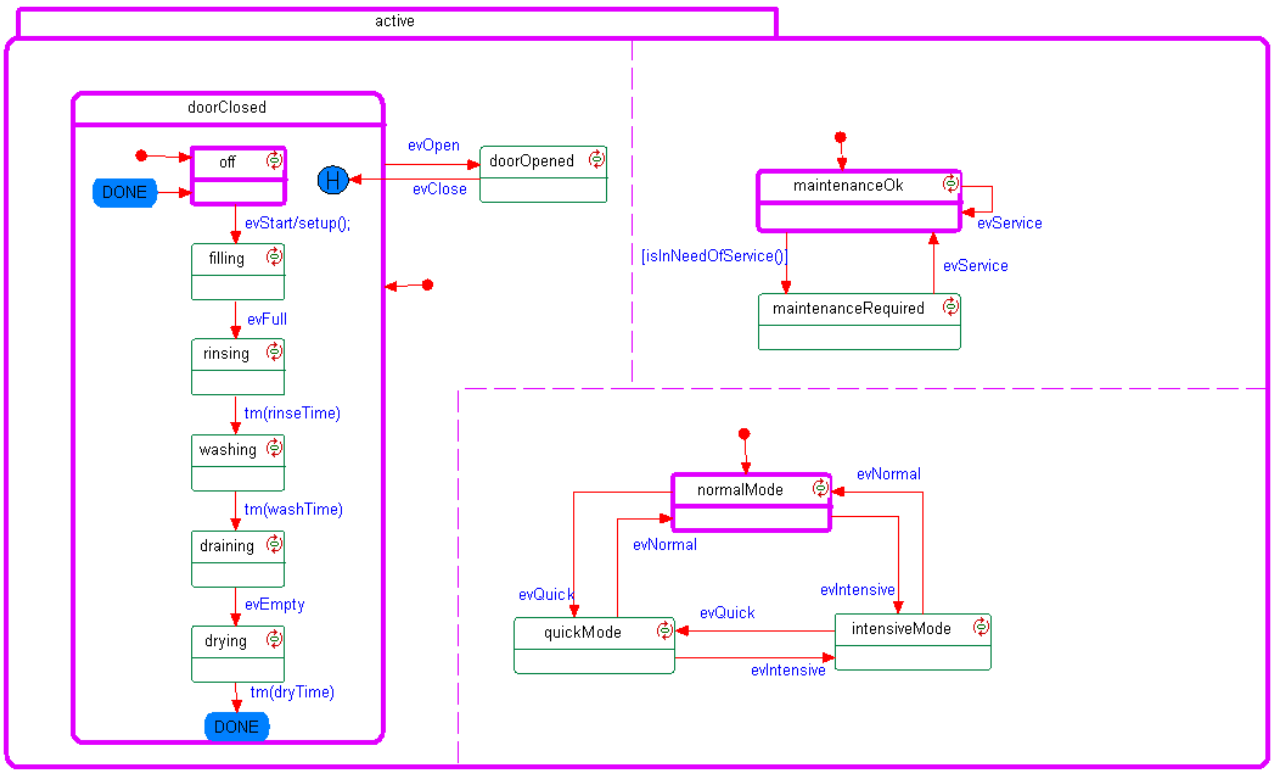

Рис. 12.21. Анимированный конечный автомат объекта класса Dishwasher

Из анимированной диаграммы на рис. 12.21 мы видим, что объект Dishwasher[0] находится в состоянии active, вложенных AND-состояниях doorClosed, normalMode и maintenanceOk (все они являются исходными). Поскольку состояние doorClosed имеет несколько OR-состояний, то объект находится в том из них, которое является исходным — off. Чтобы "посудомоечная машина" (dishwasher) заработала ее нужно включить, т. е. сгенерировать событие включения. Для этой цели используется кнопка панели анимации с изображением красной молнии. При нажатии этой кнопки откроется диалоговое окно генерации событий (рис. 12.22).

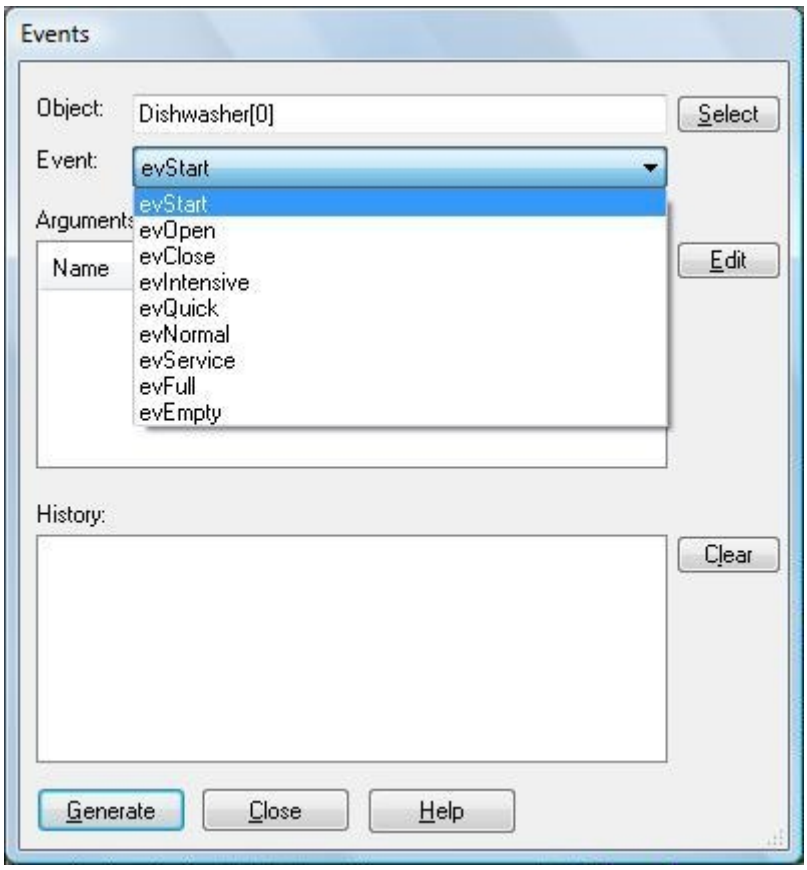

Рис. 12.22. Диалоговое окно генерации событий

В окне, представленном на рис. 12.22, необходимо выбрать объект, которому должно быть послано событие. Выбор осуществляется из списка объектов приложения, открывающегося при нажатии кнопки Select (Выбор). В дополнительном диалоговом окне нам следует выбрать объект Dishwasher[0] как показано на рис. 12.23 и нажать кнопку OK.

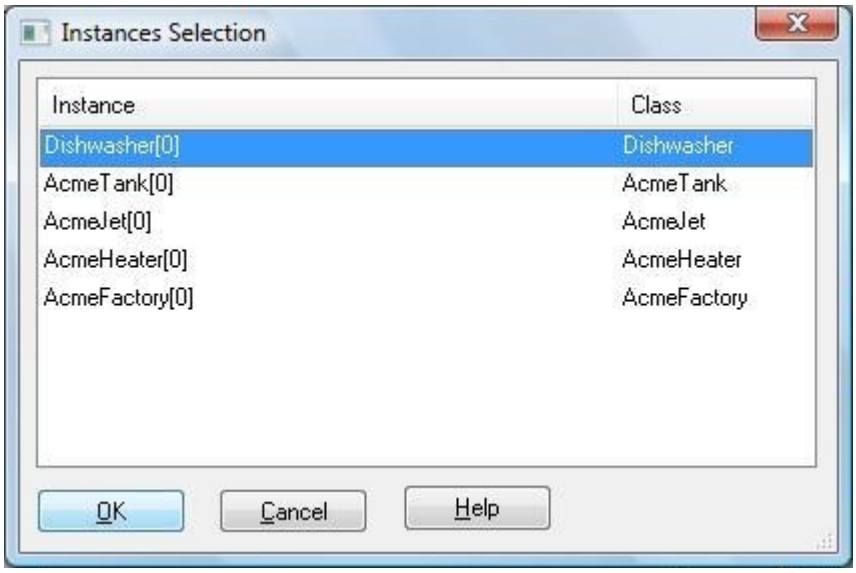

Рис. 12.23. Выбор объекта, которому следует послать событие

Затем в раскрывающемся списке Event (Событие) (см. рис. 12.22) необходимо выбрать событие для генерации. Нам необходимо выбрать evStart. Затем нужно нажать кнопку

Generate (Сгенерировать) — событие будет послано объекту. Если снова нажать кнопку Go idle, то приложение продолжит работу до тех пор, пока это будет возможно без внешних событий. В нашем случае приложение перейдет из OR-подсостояния off состояния doorClosed в подсостояние filling, затем последовательно в подсостояния rinsing, washing, draining, drying и снова вернется в подсостояние off (рис. 12.24).

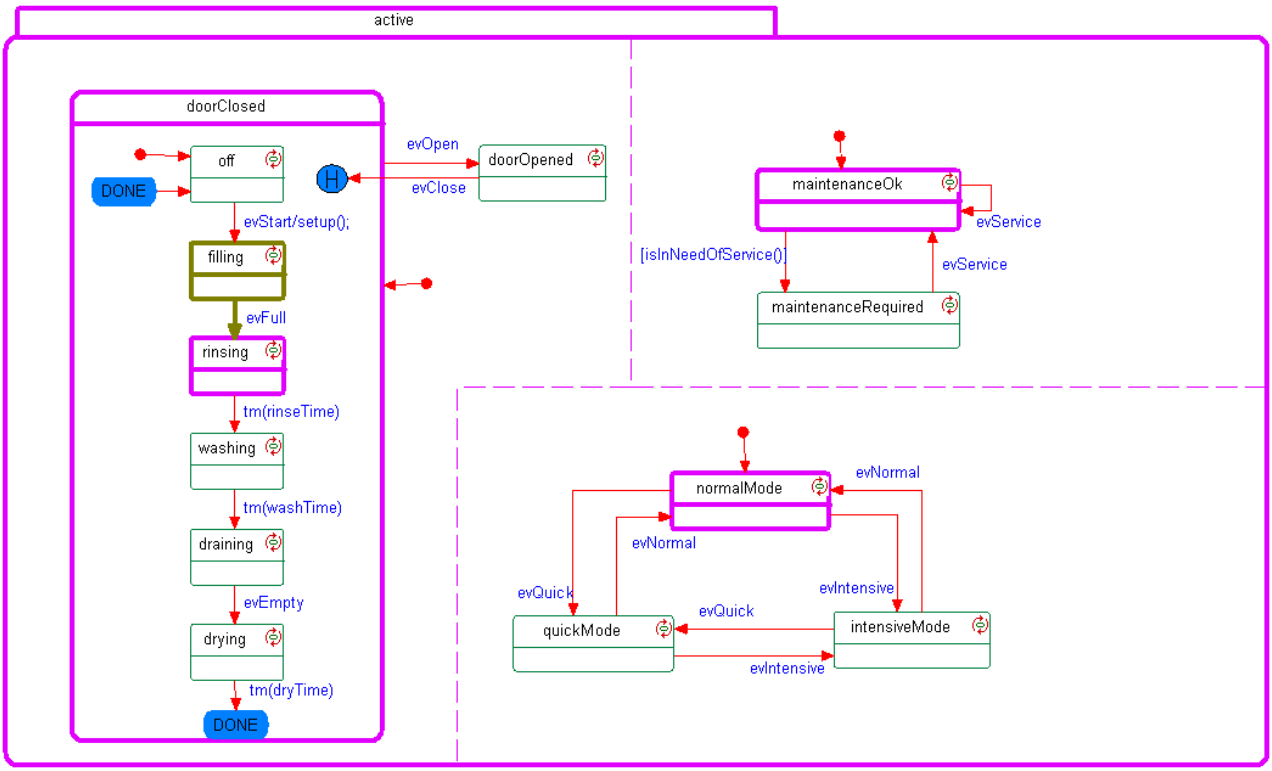

Рис. 12.24. Исполнение модели

Можно сгенерировать какое-нибудь другое подходящее событие, например, evQuick (рис. 12.25) — установить "посудомоечную машину" в режим быстрой мойки.

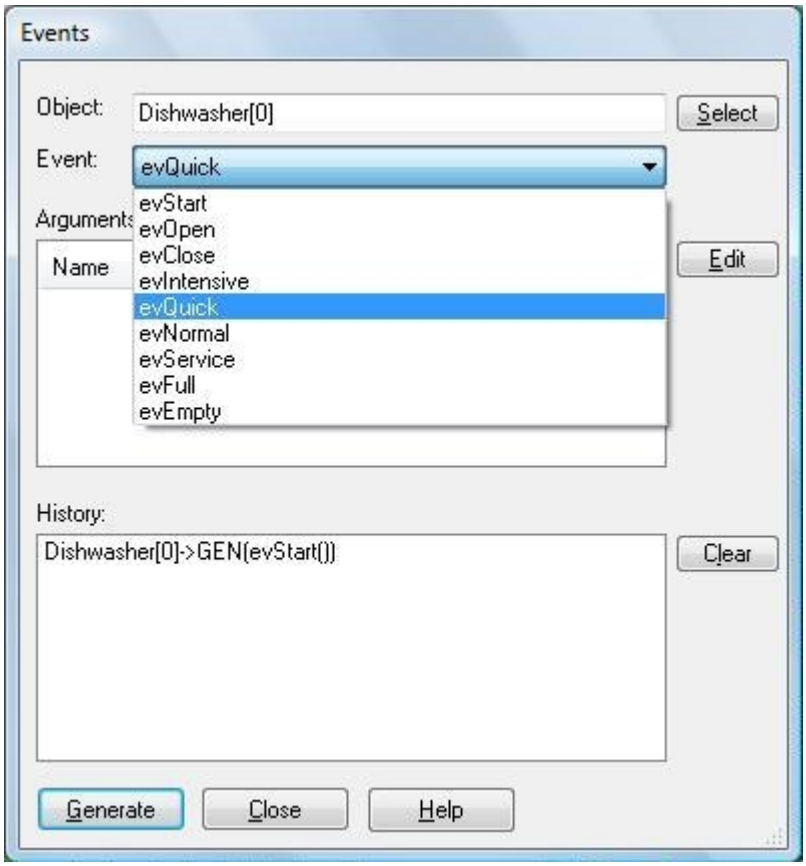

Рис. 12.25. Генерация события evQuick

При этом приложение перейдет из OR-подсостояния normalMode в OR-подсостояние quickMode (рис. 12.26).

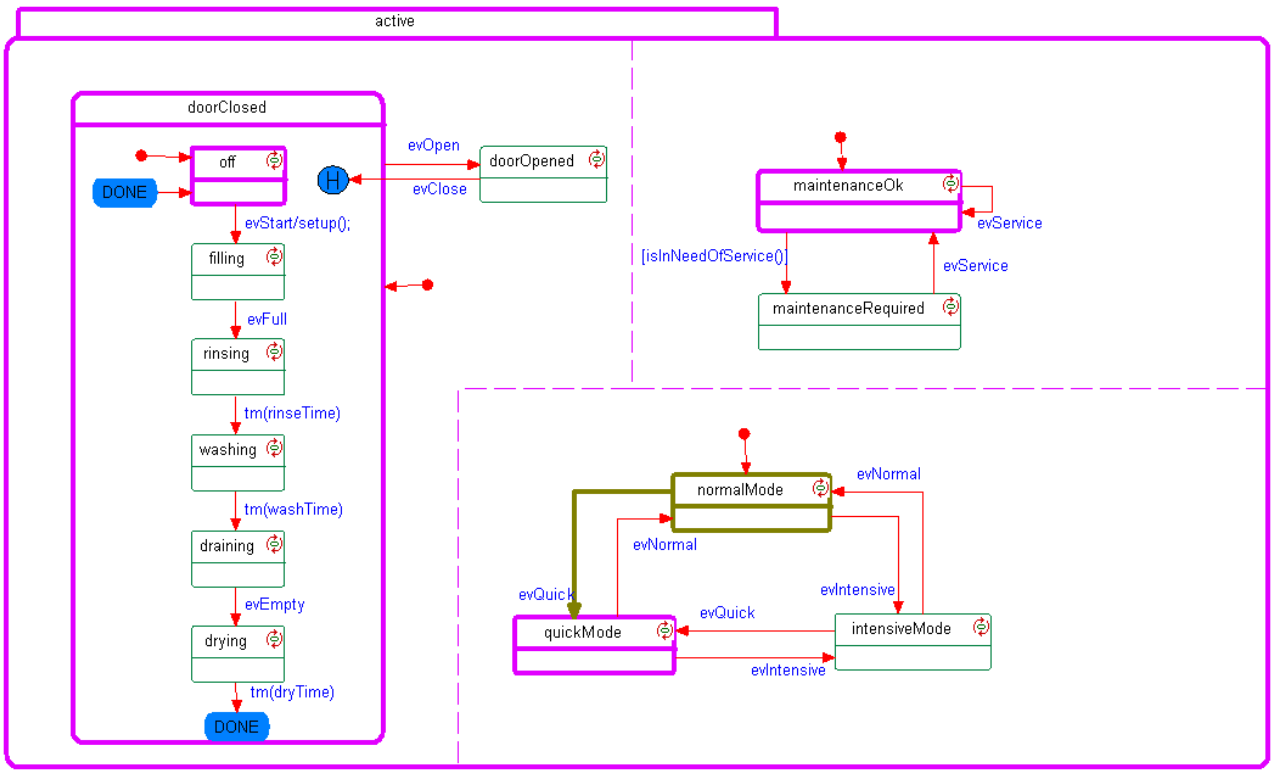

Рис. 12.26. Приложение в подсостоянии quickMode

Теперь, если мы пошлем объекту Dishwasher[0] событие evStart, то мойка будет осуществлена в ускоренном режиме.

Кроме анимированных конечных автоматов Rhapsody поддерживает анимированные диаграммы последовательности. Поскольку основное назначение обычных диаграмм последовательности заключается в спецификации порядка (протокола) взаимодействия объектов друг с другом и с внешней средой, то анимированные диаграммы последовательности позволяют после прогона анимированного приложения сравнить требуемую последовательность действий с фактической последовательностью действий. Открыть анимированный вариант можно, щелкнув правой кнопкой мыши на имени соответствующей диаграммы последовательности во время работы приложения в режиме анимации и выбрав соответствующий элемент меню (рис. 12.27).

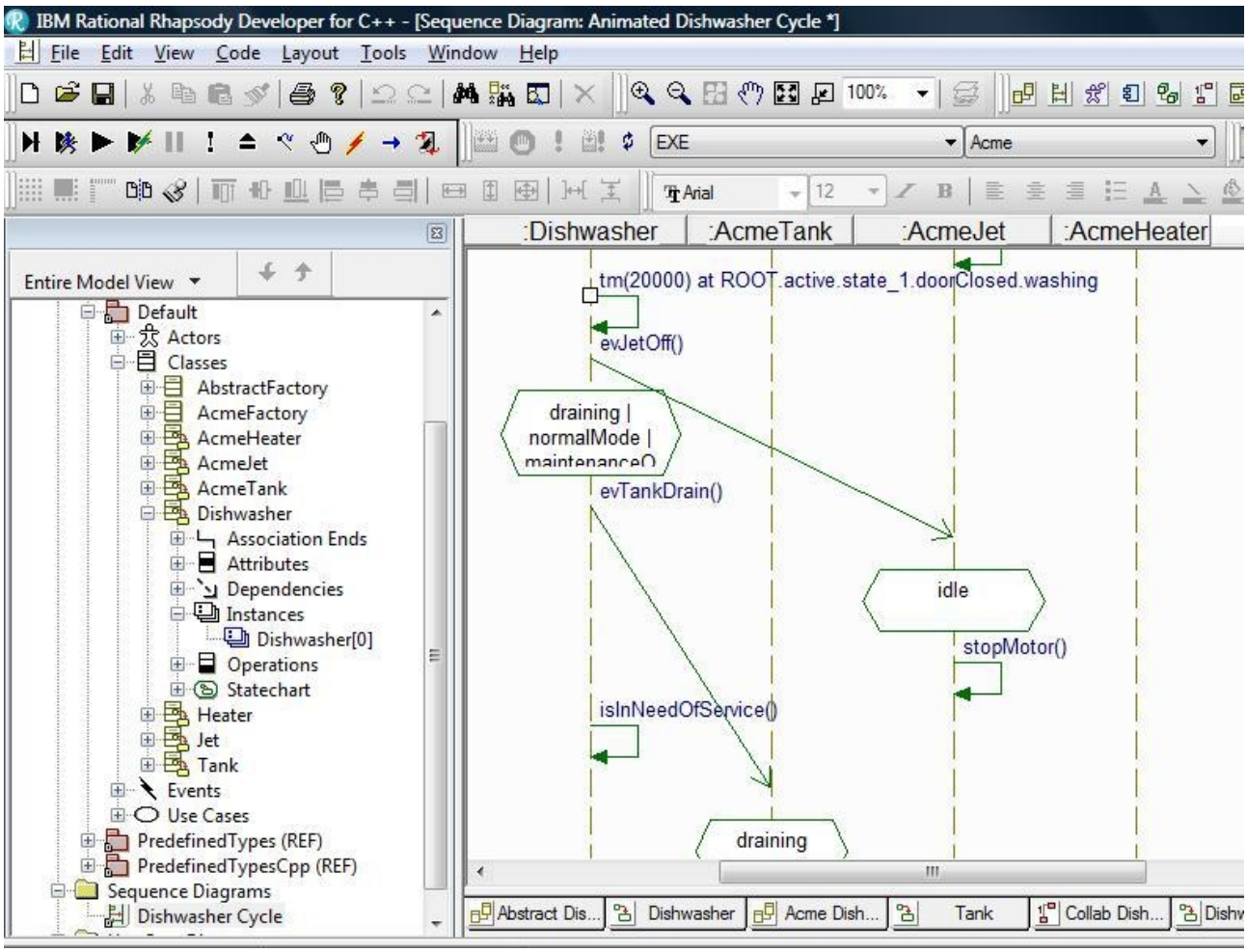

Рис. 12.27. Анимированная диаграмма последовательности

Таким образом, анимация позволяет исследовать и отладить поведение приложения на уровне UML-модели. Окончательный вариант исполняемого модуля, предназначенный для конечного потребителя, разумеется, генерируется без инструментирования.

Напомню, что ЧМИ разрабатывается с помощью других инструментов. Например, с помощью Adobe Device Central CS или QNX Photon Application Builder. Поскольку часто разработка ЧМИ выполняется параллельно с разработкой приложения, то, как мы помним, для приложения можно сгенерировать прототип ЧМИ с помощью стереотипа "Web Managed" и параметра генерации Webify.

12.6. Выводы

На этом занятии мы научились устанавливать среду разработки IBM Rational Rhapsody Developer и добавлять в нее внешний адаптер. Так же мы научились выполнять генерацию из UML-модели кода приложения, инструментировать его для поддержки анимации и запускать в режиме анимации.