УДК 681.3.06, 519.8

С.В. СЕМЕНЕЦ

СОЗДАНИЕ ПРОГРАММНОГО КАРКАСА WINDOWS CE 3.0 В СРЕДЕ RATIONAL ROSE МЕТОДОМ ОБРАТНОГО ПРОЕКТИРОВАНИЯ

1. Введение

При создании классовой модели UML целесообразно разрабатывать и использовать уже существующие модели. Они формируют базис для семейства подобных приложений - программный каркас. Согласно [1], программным каркасом называется множество предопределенных элементов модели, необходимых для моделирования систем определенного вида. Целью конкретного программного каркаса является определение архитектуры заданного класса систем либо предоставление множества повторно используемых компонентов. Программные каркасы используются в качестве шаблона при создании новой модели. В рамках данной статьи под программным каркасом (ПК) будем понимать коллекцию классов операционной системы Windows CE 3.0, используемых для проектирования и моделирования мобильных программных систем.

Использование программных каркасов позволяет значительно сократить время создания приложения. Существует также множество значительных преимуществ разработки частичного каркаса параллельно с приложением. Этот частичный каркас часто служит как неизменный абстрактный слой, который наследуют классы многих приложений.

Задача заключается в построении ПК, который позволит использовать особенности классовой структуры операционной системы при разработке архитектур приложений для Windows CE.

Актуальность создания ПК для Windows CE обусловлена, прежде всего, отсутствием такого рода каркасов, несмотря на то что, Windows CE является одной из наиболее динамично развивающихся мобильных платформ. Кроме того, в 2002 году корпорация Microsoft открыла исходные коды этой операционной системы, что дает возможность строить адекватные платформе ПК.

Целью статьи является поэтапное описание процесса построения ПК Windows CE в среде моделирования Rational Rose Enterprise Edition методом обратного проектирования (Reverse Engineering). Следует отметить, что автор не претендует на всеобъемлющую полноту полученного каркаса ввиду довольно объемного размера исходных кодов Windows CE (3Мб).

В процессе построения ПК используется метод обратного проектирования, позволяющий получить классовую структуру приложения на базе его исходных кодов.

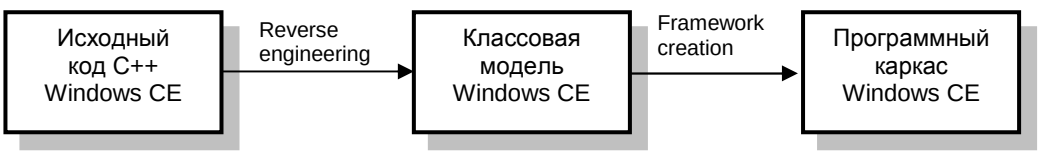

Рис. 1. Схема создания ПК Windows CE

2. Получение классовой модели ОС Windows CE

Первым шагом в решении вышеописанной задачи является получение классовой модели ОС Windows CE на базе ее исходных кодов методом обратного проектирования. Остановимся на нем более подробно.

Обратным проектированием (reverse engineering) называется процесс преобразования в модель кода, записанного на каком-либо языке программирования [2]. Результатом этого процесса является довольно большой объем информации, часть которой находится на более низком уровне детализации, чем необходимо для построения полезных моделей. В то же время обратное проектирование не позволяет полностью восстановить модель на основе кода, если инструментальные средства не включали в комментариях к исходному тексту информацию, выходящую за пределы языка реализации.

В качестве инструментального средства для обратного проектирования использовался C++ Analyzer, входящий в пакет Rational Rose. Этот продукт позволяет не только проводить обратное проектирование C++ приложений, но также используется и для управления итеративной разработкой новых приложений. Процесс получения классовой модели Windows CE можно разбить на два этапа:

1. Анализ исходного кода.
2. Экспорт извлеченной информации в модель.

Первый этап предполагает создание нового проекта в C++ Analyzer, где указываются каталоги с исходными кодами для анализа, задаются необходимые типы файлов, каталоги библиотечных и заголовочных файлов.

2.1. Анализ исходного кода Windows CE

Операционная система Windows CE – 32-разрядная, многозадачная, многопоточная, объектно-ориентированная операционная система с поддержкой фиксированной очереди приоритетов с исполнением до завершения, которая имеет открытую архитектуру. Данная ОС представляет собой набор компонентов (DLL файлы, C++ файлы), из которых, используя MS Windows CE Platform Builder, можно скомпилировать Windows CE.

Операционная система Windows CE имеет графический интерфейс, являющийся упрощенной версией интерфейса ОС семейства Windows 9x/NT/2000.

Исходные коды Windows CE содержат следующие компоненты:

Таблица 1. Компоненты Windows CE

Каталог компонента	Название	Исполняемый модуль
Private\Winceos\Comm\Apps\Rna\Remnet	remnet	Remnet.exe
Private\Winceos\Comm\Apps\Rna\Rnaapp	rnaapp	Rnaap.exe
Private\Winceos\Comm\Dhcp	dhcp	Dhcp.dll
Private\Winceos\Comm\Ppp2\Asyncmac	asyncmac	Asyncmac.dll
Private\Winceos\Comm\Tapi\Unimodem	unimodem	Unimodem.dll

Private\Winceos\Coreos\Core\Dll	Coredll	Coredll.dll
Private\Winceos\Coreos\Core\Lmem	Coredll	Coredll.dll
Private\Winceos\Coreos\Device	Device	Device.exe
Private\Winceos\Coreos\Fsdmgr	Fsdmgr	Fsdmgr.dll
Private\Winceos\Coreos\Gwe\Mgdi\Gpe	Gpe	Ddi_xxx.dll
Private\Winceos\Coreos\Nk	Nk	Nk.exe
Private\Winceos\Coreos\Nk\Kdstub	Kd	Nk.exe

В устройствах под управлением данной ОС применяются 32-разрядные RISC-процессоры Intel Strong-ARM SA-1110, NEC VR4111/4121/4122, Hitachi SH-3/SH-4, Philips PR31700, Toshiba TX 3922 и другие с тактовой частотой от 60 до 206 МГц.

Исходные коды ядра ОС содержатся в каталоге Private\Winceos\Coreos\ и соответствуют архитектуре ОС. Сервисы ядра Windows CE определяются модулем Coredll. Ядро обеспечивает базовую функциональность ОС (процессы, потоки и управление памятью) для всех устройств Windows CE, а также управление файловой системой.

Архитектура ядра Windows CE включает в себя управление приоритетами потоков, поддержку прерываний, распределение интервалов времени между процессами, что обеспечивает поддержку приложений реального времени.

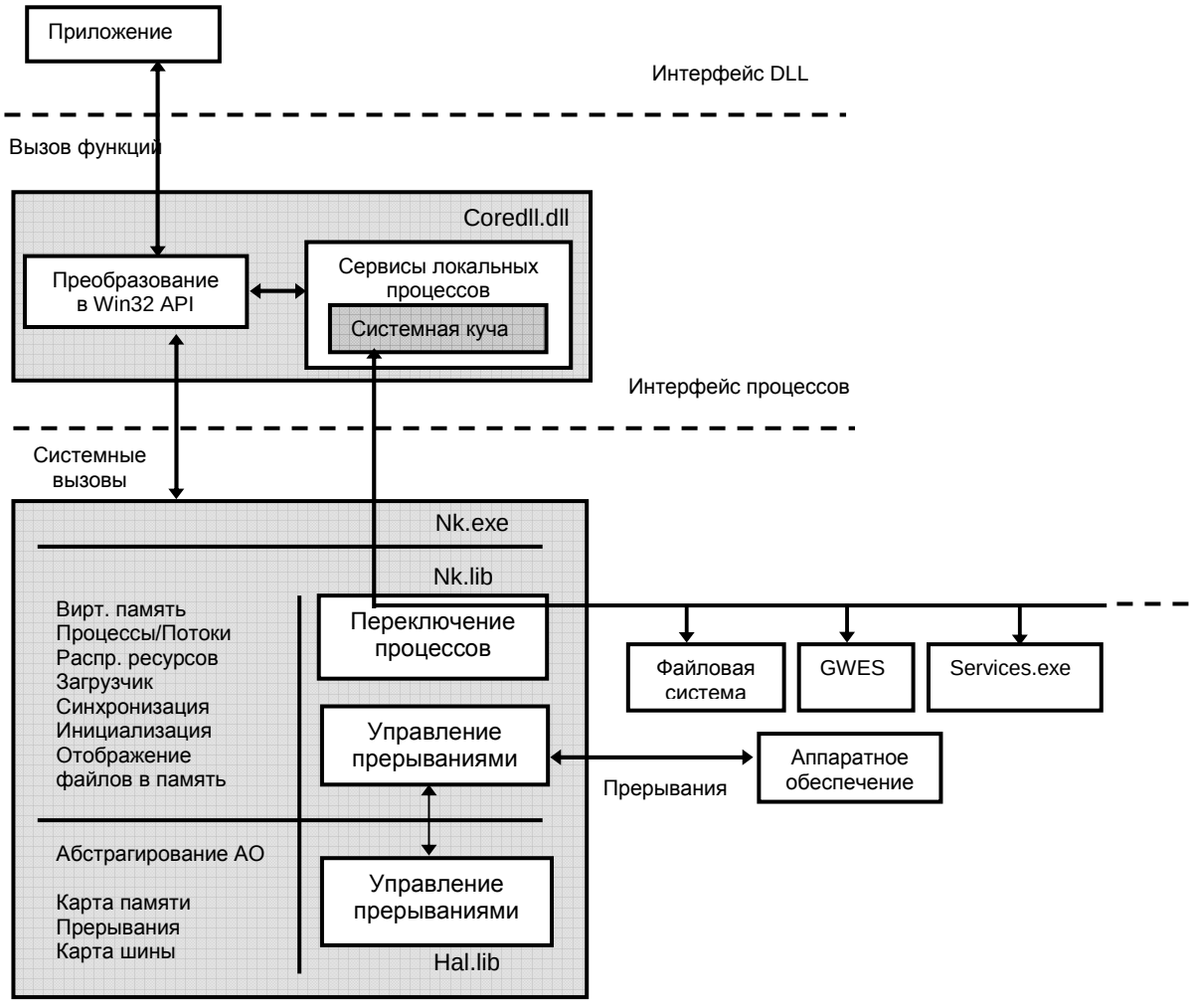

Рис. 2. Архитектура ядра Windows CE [7]

Подкаталоги Winceos\Coreos имеют следующее назначение:

Таблица 2. Функции ядра Windows CE (назначение каталогов)

Каталог	Содержимое
Core	Модуль CoreDLL – уровень интерфейса процессов
Device	Менеджер устройств Windows CE
Fsdmgr	Менеджер драйвера файловой системы
Gwe	Подсистема управления графикой, окнами и событиями (GWES)
Nk	Непосредственно ядро ОС. Nk.lib Nk.exe

На этапе анализа исходного кода Windows CE, C++ Analyzer определил 20 классовых категорий. Однако ряд совокупностей классов, определенных C++ Analyzer-ом как категории, таковыми не являются. Например, категории inc, dll, lib необходимо исключить из списка классовых категорий проекта для корректности дальнейшего анализа исходных кодов.

Помимо указания расположения каталогов и файлов для анализа, задаются также расширения анализируемых файлов и их тип: файл спецификации либо файл непосредственно тела программы. В качестве расширений файлов спецификации были заданы *.def и *.h; для файлов тела программы - *.c и *.cpp. Примечательно, что при изменении, удалении и добавлении типа анализируемых файлов динамически изменяется список анализируемых файлов, и, следовательно, требуется повторный анализ проекта.

Следует отметить, что C++ Analyzer позволяет устанавливать для каждого файла так называемый тип анализа. Это связано с тем, что некоторые исходные файлы C++ являются контекстно-чувствительными, т.е. результат их анализа не является однозначным и зависит от анализа файлов, указанных директивой #include. Для управления (координации) этой информацией C++ Analyzer может использовать три типа атрибутов анализа каждого файла из списка файлов проекта:

1. Тип 1: контекстно-независимый.
2. Тип 2: контекстно-зависимый, результаты анализа могут использоваться в проекте.
3. Тип 3: контекстно-зависимый, требуется повторный анализ каждого вхождения.

По умолчанию устанавливается первый тип атрибута анализа. Первому типу анализа файлов свойственна синтаксическая завершенность. В этом случае анализ может быть проведен в произвольном порядке, в зависимости от файлов, на которые ссылается текущий. Анализ проводится один раз с сохранением результатов в ассоциированном файле данных.

В случае, если в файле содержатся символы, определяемые в других файлах, без явного их включения, необходимо задать второй тип анализа и включить в контекст проектных файлов второго типа требуемые для анализа исходные файлы.

2.2. Генерация модели Windows CE

В целях оптимизации времени генерации в Rose предусмотрены три способа проведения реинжиниринга, каждый из которых может охватить и выполнить определенный сегмент работ. Если пользователю по каким-либо причинам не подходит ни один из трех предустановленных

способов, то Rose допускает создание собственного реверсинжениринга [4]. Стандартными способами являются:

First Look. Приближенная пробежка по телу программы.

Detailed Analysis. Детальный анализ проекта.

Round-Trip. Комбинация двух вышеперечисленных способов. Позволяет строить и перестраивать разрабатываемые приложения по принципу круговой разработки. Все настройки могут быть изменены пользователем по усмотрению. При сохранении изменений возможно указать новое имя шаблона или перезаписать уже существующее, что позволит при частом использовании обратного проектирования не терять времени на установку нужного пункта. Выбор соответствующего пункта обязательно сказывается на скорости анализа.

Генерация модели Windows CE проводилась в режиме Detailed Analysis, что обеспечило:

1. Анализ и преобразование в визуальную модель классов и структур.

2. Генерацию связей в модели (между классами или структурами).

3. Нахождение в исходном тексте комментариев и перенос их в качестве атрибутов компонентов модели.

4. Учет в проекте всех заголовочных файлов (по цепочке один за другим).

Более детальная настройка параметров анализа и экспорта возможна с помощью редактирования типа анализа.

2.2.1. Входные данные

Возможны следующие типы определения анализируемых файлов:

1. Анализ только выделенных файлов.

2. Анализ выделенных файлов и множества “include”.

3. Анализ выделенных файлов и множества “implementation”.

Под множеством “Include” понимается комбинация:

- набора файлов, указанных директивами #include;
- набора файлов, входящих в вышеуказанное множество “Include”.

Такая рекурсия позволяет задать все заголовочные, а также связанные с ними заголовочные файлы проекта.

Множество “implementation” добавляет к двум вышеперечисленным множество файлов реализации.

Существует также возможность задать, какая именно информация будет извлекаться из C++ конструкций .

Таблица 3. Исходные типы данных C++ Analyzer

N п\п	Название	Описание
1	Non-static data members	Нестатические экземпляры класса
2	Static data members	Статические экземпляры класса
3	Non-static member functions	Нестатические функции класса
4	Static member functions	Статические функции класса

5	Typedefs	Определения типов
6	Base Lists	Линейные списки
7	Instantiations	Реализации
8	Instantiation Arguments	Аргументы реализации
9	Friend Declarations	Объявления дружественных объектов
10	Template Parameter	Параметр шаблона

Определение ссылочных типов классов допускается следующими способами:

1. Поиск определений производится в соответствии с типом анализируемых файлов.

2. Поиск в множестве #include.

3. Поиск в множестве #include, однако в случае неудачи производится поиск по всем файлам проекта.

4. Поиск в множестве #include, в случае неудачи включается множество базовых файлов.

2.2.2. Выходные данные

Вкладка Output позволяет определить заголовок, отображаемый на каждом окне диаграммы, путь выходного файла модели, вид нотации (UML, Booch, OMT), параметры классовой модели, такие, как:

– создание диаграмм классов для каждой категории;

– создание категорий классов;

– включение подсистем в генерируемую модель.

Существует также возможность задать критерий ассоциации классов с определенными категориями, такими, как аннотация класса, проект либо определение, содержащиеся в каталоге класса. Кроме того, на вкладке определяется политика по отношению к модулям (Units) – метод их создания в выходной модели и ассоциации с категориями.

2.2.3. Типы классов

Для генерации модели Windows CE были включены все типы классов с целью наиболее полного отражения специфики операционной системы:

1. Public.

2. Protected.

3. Private.

4. Implementation.

Аналогичные типы были определены для шаблонов, определений и списков. Фундаментальные типы, такие как int, char и т.д., в классовую модель не включались, как и реализации шаблонов.

2.2.4. Отношения

В рамках анализируемых кодов Windows CE как статические, так и нестатические атрибуты, представлялись как ассоциации. Анализу, с точки зрения отношений, подвергались только классовые типы Public, Protected, Private, Label. В силу того, что C++ Analyzer не может определить кратность отношения по определению указателя в исходном коде, для анализа была выбрана кратность по умолчанию, равная 1 – 0..1.

Критерии создания отношений использования позволяют определить, какие структуры данных будут использоваться для вышеозначенных отношений. Однако при построении модели отношения использования не применялись, так как не представляется возможным построить однозначное соответствие между этим видом отношений и предопределенными структурами данных .

Отношения наследования строились для всех базовых классов и их подклассов, включая типы Public, Protected и Private.

В модель также были включены отношения реализации между экземплярами и шаблонами объектов.

2.3. Экспорт модели в Rational Rose

В соответствии с вышеописанными настройками производится экспорт модели в формат приложения Rational Rose – MDL файл. Размер файла модели 2,49 Мб. Файл сохраняется в заранее определенном в C++ Analyzer месте. Далее с ним можно работать, как с обычной моделью Rational Rose.

3. Создание программного каркаса в Rational Rose

Под программным каркасом (ПК) будем понимать коллекцию классов, используемых в нескольких различных приложениях. Часто классы внутри каркаса взаимосвязаны. Они могут быть абстрактными и использоваться через наследование. Интерфейс прикладного программирования (API), реализованный в Java, является примером каркасных пакетов.

Все программные каркасы хранятся в отдельном каталоге \Framework\Frameworks, в инсталляционном каталоге Rational Rose. Следует помнить, что все пакеты, определенные в ПК, хранятся как управляемые модули в отдельном файле. Для доступа к содержимому пакета в ПК необходимо загрузить соответствующий управляемый модуль.

ПК в Rational Rose – это множество элементов модели, необходимых для моделирования систем определенного вида. Целью специфицирования ПК может быть как определение архитектуры систем, так и предоставление множества компонентов повторного использования. ПК используются как шаблоны при создании новой модели [1].

Каждый пакет программного каркаса хранится в виде управляемого блока в отдельном файле. Для доступа к содержимому пакета программного каркаса необходимо загрузить соответствующий управляемый блок (File >Units > Load).

ПК определяется следующими файлами:

1. Файл ИмяКаркаса.mdl, который содержит базовую модель ПК.

2. Опциональный файл ИмяКаркаса.ico, который включает иконку, символизирующую ПК в диалогом окне Create New Model. Если .ico файл отсутствует, Rational Rose отображает иконку по умолчанию.

3. Опциональный файл ИмяКаркаса.rtf, который содержит описание ПК, отображаемое в диалогом окне Create New Model, если пользователь нажимает кнопку Details. В случае отсутствия файла описания, отображается текст описания по умолчанию.

4. Опциональный файл параметров вызова, содержащий имя диаграммы, которая открывается при создании модели с использованием ПК.

Полученная модель Windows CE 3.0 содержит около 800 классовых структур, объединенных в 13 следующих пакетов: unspecified, arm, dll, gpe, inc, kdstub, kernel, lib, mips, ppc, x86, nkproof, lmem (рис. 3).

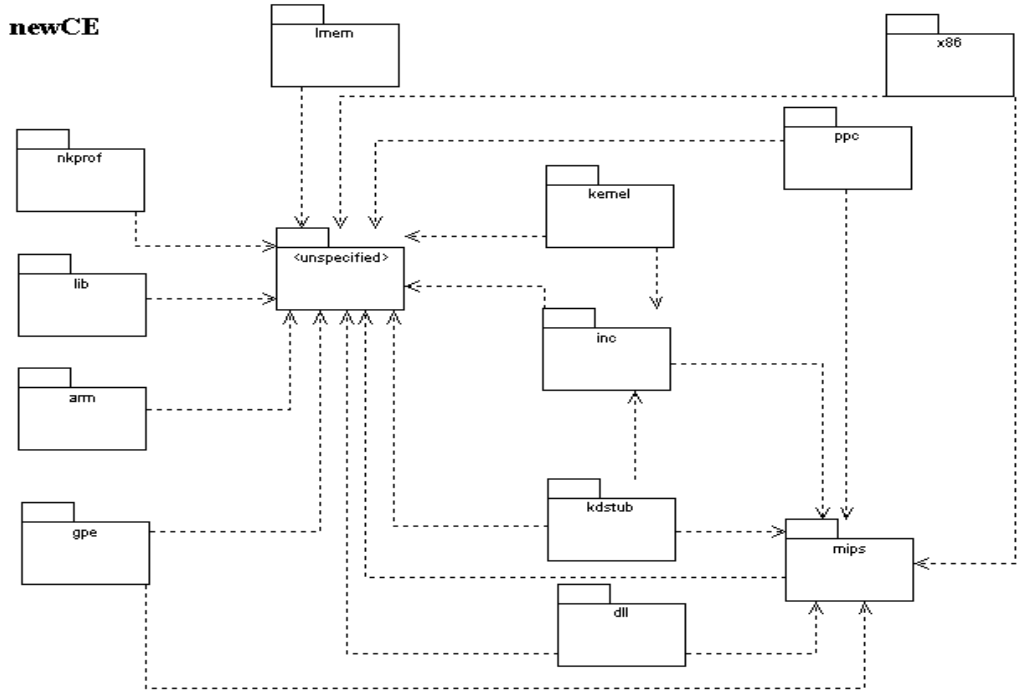

Рис. 3. Диаграмма пакетов модели Windows CE

Модель была доработана с целью приведения к читаемому виду: исключены классовые структуры, в которых нет необходимости, разрешены несвязанные либо некорректно связанные структуры, разрешены отношения компонентов и пакетов.

При создании ПК было получено сообщение о наличии двух ошибок: “Invalid procedure call argument” и “Automation Error”, вызванных конфликтом доступа к модели, открытой в Rational Rose и одновременно примененной в качестве программного каркаса.

Полученный ПК показан на рис. 4.

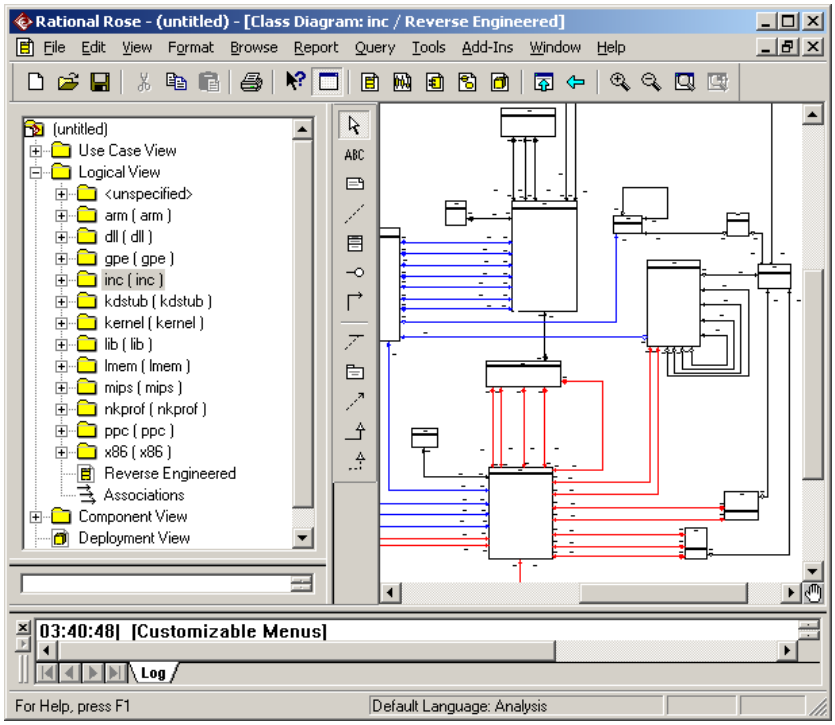

Рис. 4. Программный каркас Windows CE

4. Создание приложений для Windows CE

Особенности создания приложений для Windows CE. Несмотря на то, что ОС Windows CE основана на Win32 API (по сути, WinCE API является подмножеством Win32 API), существует множество особенностей, отличающих ее от традиционных приложений Windows. Остановимся на наиболее существенных из них.

Поскольку для хранения данных Windows CE использует память, содержимое файловой системы находится в RAM. Операционная система, а также все приложения, необходимые для работы с устройствами, хранятся в ROM. RAM в Windows CE разделена на две части: память программ, хранилище объектов. Хранилище объектов содержит файловую систему и системный реестр. Кроме того, оно содержит БД Windows CE, пользовательские БД (например, БД контактов) и БД, созданные приложениями.

Одной из отличительных особенностей приложений для мобильных систем является, как правило, необходимость синхронизации данных с настольной системой. Для этих целей в Windows CE используется технология ActiveSync, которая позволяет при создании приложений предоставлять сервисы для синхронизации данных этих приложений с настольными системами.

Для отладки приложений под Windows CE используется либо эмулятор WinCE на настольной системе, либо средства удаленной отладки приложений на мобильной системе (Remote API). В обоих случаях разработчик должен указать конкретный тип процессора мобильной системы. Скомпилированное приложение необходимо переписать на мобильное устройство, для

чего следует воспользоваться вышеупомянутым ActiveSync. WinCE API включает в себя интерфейсы для отладки приложений (такие, как DebugActiveProcess и DebugEvent).

Необходимо отметить уменьшенный, по сравнению с Win32, набор стилей окон (window styles), сообщений между окнами (window messages), используемых цветов и шрифтов. Кроме того, вследствие использования в КПК специальных средств ввода информации (клавиатура в большинстве из них отсутствует), таких, как стило, кнопки навигации, голос, в WinCE API используются специальные расширения, некоторые из которых заменены соответствующими элементами Win32 (например, command bar).

Ограничения касаются также и управления исключениями. Несмотря на то, что Win32 поддерживает управление структурированными исключениями, WinCE не имеет такой возможности.

Стандартная библиотека MFC (Microsoft Foundation Class Library) также отличается от MFC для Windows CE. Значительные отличия можно заметить как среди наличия определенных классов, так и среди возможностей, поддерживаемых каждым классом. Кроме того, некоторые классы в MFC для Windows CE являются уникальными для этой платформы.

При создании приложений для Windows CE следует обратить внимание на значительно меньший объем RAM, нежели в настольном ПК. Приложение должно использовать как можно меньший объем памяти и обладать возможностью взаимодействия с системным менеджером памяти [3].

Преимущества использования программного каркаса.

1. Наглядность структуры операционной системы (рис. 4).

Наличие программного каркаса Windows CE дает возможность на любом этапе проектирования использовать внутренние структуры операционной системы на любом уровне, вплоть до конкретных операций, атрибутов и соответствующих им описаний Microsoft, которые извлекаются из комментариев во время обратного проектирования (рис. 5).

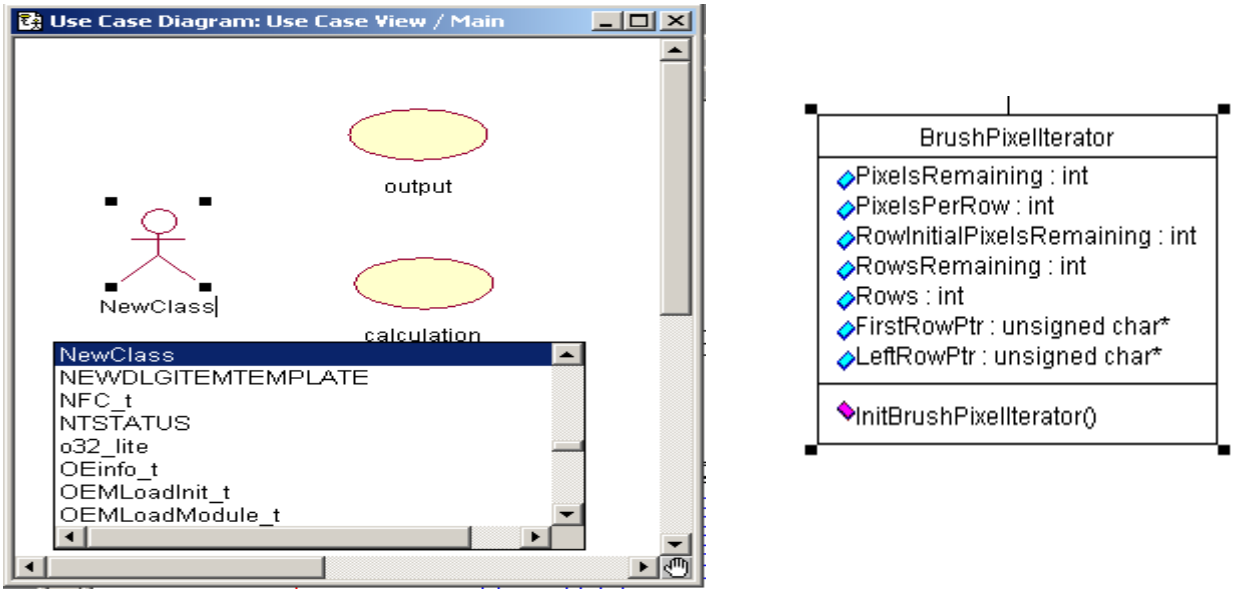

Рис. 5. Преимущества ПК: автоматическая подстановка, операции и атрибуты

2. Автоматическая подстановка возможных имен структур, классов, пакетов, компонентов (рис. 5). Построение модели приложения для платформы Windows CE на базе вышеописанного программного каркаса позволяет автоматически, при именовании элементов модели, осуществлять выбор из уже имеющихся названий компонентов, хранящихся в базе программного каркаса.

3. Построение отношений зависимости, обобщения и ассоциации на базе имеющейся классовой структуры ОС Windows CE.

5. Выводы

Одной из основных задач в процессе создания ПК была генерация базовой модели на основе исходных кодов Windows CE. Однако, благодаря гибкости инструментального средства Rational C++ Analyzer, она была успешно решена.

Другая важная задача – доработка полученной модели до читаемого вида. Задача была затруднена наличием довольно большого числа объектов в полученной модели и отсутствием средств автоматизации для решения такого рода задач.

К недостаткам разработки данного ПК можно отнести то, что существуют более поздние версии ОС, однако их исходные коды закрыты разработчиком. Кроме того, необходимо отметить, что использованные исходные коды Windows CE 3.0 являются неполными, что, с одной стороны, ускорило разработку ПК, но с другой, – ограничило модель ОС.

В целом задача построения программного каркаса операционной системы Windows CE методом обратного проектирования решена успешно, что открывает перспективу построения комплексной системы создания приложений для мобильных систем как на Windows CE, так и на других платформах, с доступными исходными кодами.

Темпы развития рынка мобильных устройств на сегодняшний день приближаются к темпам развития рынка персональных компьютеров. Их распространение с каждым днем находит место в различных областях науки и техники, причем производительность таких устройств в определенных приложениях сравнима с производительностью персональных компьютеров. Следует отметить существенное различие как в задачах персональных и мобильных устройств, так и в механизмах решения этих задач (и аппаратных, и программных). Однако, несмотря на столь существенную разницу, многие технологии программной инженерии могут быть перенесены или адаптированы к мобильному программному обеспечению. Для успешного решения этой задачи необходимо учитывать архитектурные особенности мобильной платформы. Именно с этой целью и создан программный каркас операционной системы Microsoft Windows CE, что позволяет использовать уже имеющиеся архитектурные решения при построении приложений.

СПИСОК ЛИТЕРАТУРЫ

1. Rational Rose Enterprise Edition 7.5.0103.1920 Help Reference // Copyright (C) 1991–2001 Rational Software.
2. Booch, Rumbaugh and Jacobson The Unified Modeling Language User Guide. – Addison-Wesley: 1999. – P. 30 – 31.
3. Семенец С.В., Гавсиевич И.Б. Обзор программных архитектур ОС мобильных платформ // Математичні машини і системи. – 2003. – № 1. – С.119 –134.
4. http://www.interface.ru/fset.asp?Url=/rational/ross2/ross2.htm «Rational Rose для разработчиков».