Сергей Орлов

Компонентное ПО и объектная прикладная среда Oberon

Источник: ComputerWeek-Moscow, 1995, # 37.

Объектно-ориентированные языки, среды разработки, операционные системы и пользовательские интерфейсы все глубже проникают в нашу жизнь, становясь неотъемлемой частью современной вычислительной среды. Работы в этом направлении ведутся уже давно. Среди последних достижений достаточно упомянуть технологию составных документов и компонентного программного обеспечения OpenDoc компаний Apple, IBM и Novell или технологию объектной компоновки OLE 2.0 корпорации Microsoft. Разработчики с успехом используют объектно- ориентированные методы для создания приложений и систем. Объектные принципы применяются в большинстве современных языков программирования.

В последнее время эти возможности все полнее стали отражаться и в операционных системах, что наглядно проявляется в интерфейсах последних версий ОС — Windows 95, OS/2 Warp и в операционной среде Macintosh. Крупные американские производители ПО, ведущие интенсивные разработки в этом направлении, пристально следят за успехами конкурентов. Пресса подробно освещает их деятельность и всесторонне обсуждает каждое новшество компаний-гигантов. Между тем исследования в области подобных современных технологий ведутся не только в США, но и в Европе. Эти работы часто ни в чем не уступают достижениям известных компаний, но по понятным причинам им уделяется гораздо меньше внимания. Настоящая статья представляет собой попытку хотя бы частично восполнить указанный пробел, рассказав об одном из интересных проектов, который может внести вклад в развитие технологии компонентного ПО. Данный вид программного обеспечения уже в ближайшем будущем может решающим образом изменить характер вычислительного процесса, программирования приложений, архитектуру операционных систем и даже методы использования программного обеспечения. Немного истории: от объектно-ориентированного к компонентному ПО Термин "объектная технология" применяют сегодня к ПО самого различного типа — от операционных систем до инструментальных средств разработки. Что же дают объектные методы пользователям и разработчикам? В объектно-ориентированном программировании после определения и написания программного кода объекта его можно повторно использовать в любой другой системе. Так как новый код требуется только для того, чтобы связать объекты, объем программирования, необходимый для разработки новой системы (а следовательно, время и ресурсы, которые нужны для ее реализации), значительно сокращается. Более того, поскольку объектные системы отличаются модульностью, они легко переопределяются или модифицируются программистом в соответствии с изменениями в характере деятельности компании. Объекты можно просто "отключить" и задействовать вместо них другие. К тому же, если при работе над программным проектом логика некоей самостоятельной задачи инкапсулируется в объект, это сразу освобождает других программистов от необходимости заниматься ею. Разработчикам нужно быстро и эффективно создавать специализированные приложения, отвечающие конкретным требованиям, что подталкивает их к переходу на объектно-ориентированную модель. Объектные методы — это один из этапов в эволюции информатики. Они позволяют создавать формируемые из компонентов приложения с согласованными интерфейсами и повторно используемым кодом.

Однако данная технология предъявляет очень высокие интеллектуальные требования к разработчикам. Ее нелегко освоить тем, кто воспитан на традиционных подходах. Кроме сложности данной технологии, некоторые аналитики отмечают и такой ее недостаток, как высокая стоимость обслуживания приложений, содержащих объекты. Что касается другой категории (гораздо более многочисленной, чем разработчики) — пользователей ПО, то их в основном интересует качество приложений, с которыми они имеют дело, и способность последних сделать работу продуктивнее. Пользователям не так уж и важно, являются те объектно-ориентированными или нет.

Все это заставило искать новые решения, совершенствовать существующие методы и в итоге привело к возникновению технологии компонентного программного обеспечения. В отличие от объектных методов здесь в центре внимания находятся именно конечные пользователи. Как же с помощью компонентов удается достичь того, чего невозможно получить с помощью объектов? Прежде всего следует уяснить себе основные различия между объектами и компонентами. Объект — это фрагмент программного кода или спецификации, которую можно использовать для построения приложения. Компонент же — не просто спецификация. Это готовый к работе программный модуль, представляющий непосредственную ценность для конечных пользователей.

Для сборки приложений из подобных компонентов разрабатываются специальные технологии, такие как OLE, COM и OpenDoc. Интересным применением технологии компонентного ПО стали компонентные прикладные среды, позволяющие свести к минимуму сложности освоения библиотек классов и реализовать повторное использование крупных программных фрагментов, уже имеющих полезные потребительские свойства. Одной из таких прикладных сред (с собственным языком программирования) является не очень известная, но явно заслуживающая внимания система Oberon.

Проект Oberon

Компонентное программное обеспечение нуждается в совершенно новом подходе к созданию программных средств. Такое ПО должно обеспечивать расширение и включение в него новых частей на mb /% выполнения. Это требует решения достаточно сложных проблем (таких, как расширение или создание подклассов компонентов выполняемой системы), с которыми связан целый ряд вопросов наследования, делегации, передачи сообщений и агрегирования.

Необходимо определить не только, каким образом компоненты будут создаваться с помощью языка программирования, но и как они будут комбинироваться друг с другом, образуя единую систему. Один из возможных путей состоит в своеобразном симбиозе языка программирования и операционной системы, образующих единое целое.

Такой подход дает целый ряд преимуществ, и именно он был заложен в основу проекта Oberon Института компьютерных систем (Institute for Computer Systems ETH, Цюрих, Швейцария). Oberon — это одновременно и среда (система) программирования, и язык. Объектно-ориентированный язык Oberon, созданный Никласом Виртом (Niklaus Wirth) в 1986 г, по существу, явился развитием таких языков программирования, как Algol, Pascal и Modula-2. Создателями системы Oberon стали Вирт и Юрг Гуткнехт (Jurg Gutknecht) из ETH. В 1989 г. эта работа завершилась: система и язык Oberon были реализованы в таких средах, как DOS, Windows и Unix. За Oberon последовала новая версия — языковой процессор Oberon-2. Oberon-2, представляющий собой строго типизированный компилируемый язык, поддерживает модульное и объектно-ориентированное программирование. В это ПО входит компилятор и отладчик (хотя большинство ошибок работающие с Oberon программисты могут выявить еще на этапе компиляции). Проверка предусловия и постусловия выполняется в Oberon с помощью оператора ASSERT (аналогично языку Eiffel). Целостность памяти обеспечивает включенный в него "сборщик мусора" (довольно нестандартное решение для компилируемых языков).

В 1991 г. система Oberon была усовершенствована и стала больше походить на коммерческое компонентное ПО, но осталась при этом достаточно понятной, гибкой и компактной. В очередной версии, получившей название Oberon System 3, к классической реализации Oberon были добавлены новые концепции. Например, здесь появилась библиотека объектов, которая хранится в файле и допускает включение в нее новых объектов. Она функционирует как контейнер объектов и позволяет приложениям обращаться к ним с помощью простого механизма индексирования, напоминающего доступ к элементам массива. Управлять доступностью и совместным использованием объектов можно посредством частных и общих библиотек. Даже шрифты в Oberon System 3 представляют собой библиотеки объектов — шаблонов символов, индексируемых по значению кода ASCII, а текст, по сути, сводится к потоку пар (библиотека, индекс).

В тексте могут быть и иные объекты с произвольными связями (указателями на другие объекты), которые часто хранятся в той же библиотеке. Если одна библиотека содержит ссылки на другую, то в этом случае она импортирует из нее соответствующий объект. Для таких ссылок применяются обычные переменные-указатели, а для хранения имен объектов — механизм словаря. Общие библиотеки и совместно используемые объекты позволяют связывать объекты с различными документами, что немедленно вызывает ассоциации с известными объектными технологиями — OLE и OpenDoc. Объекты имеют атрибуты (в Oberon System 3 они называются локальными состояниями) и могут реагировать на сообщения, что в целом соответствует принципам, заложенным в основу таких языков, как Visual Basic, ObjectPAL, и оболочек объектно-ориентированных ОС. С помощью специального сообщения пользователь может управлять состоянием объекта — получать и устанавливать атрибуты. Специальный редактор (Inspector) позволяет редактировать их. В Oberon System предусмотрено три класса сообщений. На один из них реагируют все объекты, на другой — только визуальные. Третий класс сообщений определяется самим приложением. В целом можно сказать, что свободно распространяемое ПО Oberon System 3 и проще, и, возможно, более логично построено, чем другие коммерческие объектно-ориентированные среды. Оно приближается к последним по основным характеристикам, но намного гибче, компактнее и в общем доступнее для понимания. Его

"некоммерческая" родословная невольно ассоциируется с судьбой столь хорошо известной сегодня ОС Unix, некогда возникшей как чисто экспериментальная система. Кто знает, вдруг и Oberon ждет такое же будущее?

System 3 послужила основой для дальнейших исследований и разработок, которые привели к созданию в 1994 г. системы Oberon/F (в виде бета-версии). Она будет реализована на нескольких платформах (пока это Windows и Macintosh System 7). Намечено также выпустить версии для OS/2 и Unix/Motif. Oberon/F — разработка компании Oberon Microsystems (Цюрих), которую возглавил Н. Вирт. В отличие от Oberon System 3 система Oberon/F стала коммерческой и стоит 350 долл. (учебная версия предлагается бесплатно). Oberon/F System — новое пополнение в семействе объектно-ориентированных прикладных сред (таких, как нынешняя AppKit в NextStep и будущая TalAE в Taligent). Как и Oberon System 3, Oberon/F — это и объектно-ориентированный язык разработки, и среда программирования. Она представляет собой "прослойку" объектных средств, работающую поверх операционной системы и позволяющую писать переносимые и дополняемые приложения. По существу, Oberon расширяет возможности базовой операционной системы, превращая ее в полноценную объектно-ориентированную среду. Основное ее назначение состоит в том, чтобы обеспечить повторное использование программного кода — взаимодействующих друг с другом компонентов, которые можно было бы применять при проектировании и расширении программного обеспечения. Система Oberon предоставляет такие средства, как библиотеки совместно используемых объектов и реквизиты (повторно используемые объекты, реализующие элементы GUI-интерфейса). Здесь также существует три класса сообщений, а связи между объектами можно определять на этапе выполнения. Объекты могут выводиться на экран и взаимодействовать с другими объектами. Процесс разработки состоит в интерактивном проектировании GUI-интерфейса и программировании фрагментов кода для работы с реквизитами (приверженцам Visual Basic такой подход покажется знакомым). Реквизиты В основе GUI-интерфейса Oberon System лежит модель реквизитов. Реквизит (диалоговый элемент) - это объект, встраиваемый в любой пользовательский интерфейс (например блок списка). Реквизиты могут включаться в текстовый документ, диалоговое окно, графический редактор и т. д. Все реквизиты разрешается интегрировать и повторно использовать в среде Oberon на любой поддерживаемой платформе. При этом реквизит-контейнер обычно управляет дочерними реквизитами. Базовыми контейнерами являются панельные реквизиты (обычные окна редактирования) и текстовые реквизиты (полноценные текстовые редакторы с поддержкой встроенных объектов). На этой основе можно строить и более сложные контейнеры, имеющие вид составных документов или структурированных меню (с помощью более специализированных элементов). Для большинства контейнеров определен набор правил взаимодействия, и поэтому допускается включение практически любых реквизитов во все контейнеры. Концепция контейнеров позволяет строить иерархические структуры со сложным включением. Реквизиты можно модифицировать и использовать независимо от того, где они располагаются.

При разработке приложения необходимо указать выполняемые реквизитами действия (например, в ответ на такое событие, как щелчок мышью). Разработчик может связать действие с любым реквизитом при помощи редактора Inspector. Алгоритм этого действия записывается в виде процедуры на языке Oberon. Процедура выполняет поиск объектов в интерфейсе пользователя и изменяет их состояние. Аналогично прикладной среде Model Viewer Controller (MVC) языка Smalltalk, которая также дает возможность минимальным образом "привязывать" приложения к их интерфейсу, в Oberon в роли абстрактных типов общих структур данных часто выступают специальные атрибуты. Их можно интерактивно скомпоновать с пользовательским интерфейсом (UI) и визуализировать с помощью ряда других его элементов. Подобный механизм "слабой привязки" и другие методы позволяют сделать UI до такой степени независимым от приложения, что его можно модифицировать или даже расширять во время выполнения приложения.

Сам экран также служит объектом, куда входят реквизиты-документы, которые, в свою очередь, состоят из меню и всевозможных полей и т. д., вплоть до таких простейших реквизитов, как командные кнопки. Система Oberon предоставляет графический редактор, работающий с окружностями, эллипсами и другими реквизитами. Oberon имеет достаточно высокие скоростные характеристики: вы можете легко редактировать экран, содержащий тысячи реквизитов.

Составные документы

Oberon — прикладная среда, ориентированная на взаимодействие с документами. Все объекты здесь являются документами, и вы можете редактировать их на этапе разработки системы. Фактически построение интерфейса сводится к редактированию документа. Каждый документ Oberon содержит одно или более представлений - программных компонентов (реализуемых в виде отдельных модулей), благодаря которым вы можете просматривать тот или иной тип данных (например, текст, графику или даже электронную таблицу). Это очень характерная деталь: аналогичные принципы реализуются и в технологии OpenDoc, позволяющей собирать

компоненты в стандартный документ, который "знает", как с ними работать. Составные документы OpenDoc могут содержать информацию, создаваемую пользователями в различных приложениях. Например, текстовый процессор создает текст, графическая программа — изображение, а приложение БД — форму базы данных. Ко всем этим компонентам можно обращаться из составного документа. Таким образом, документ содержит указатели на текстовые объекты, изображения, шрифты, объекты данных и другие приложения. Если составной документ связать с другими документами, то они при внесении в них изменений будут автоматически обновлять друг друга. В системе Oberon пользователи могут создавать новый интерфейс или модифицировать существующий, прибегнув к обычной буксировке мышью. Такой пользовательский интерфейс (UI) "фиксируется" только после явной блокировки с помощью редактора Inspector, но пользователи для внесения изменений и настройки могут в любое время его разблокировать. (Подобное динамическое поведение требует относительно "тонкого" взаимодействия между UI-интерфейсом и приложением.) На этапе выполнения вы можете также определять связи между объектами и просматривать их, задействовав Inspector. Сохранение всех участвующих в такой структуре (напоминающей сеть) объектов в библиотеке дает возможность зафиксировать данную структуру (сделать ее постоянной). Сформированные связи позволяют транслировать сообщения между объектами и образуют основу для организации всего визуального интерфейса. Данную схему невозможно было бы эффективно реализовать без строгого протокола обмена сообщениями, который обеспечивает полную интеграцию реквизитов.

Модули

Система Oberon структурирована как дерево модулей в разделяемом адресном пространстве. Эти модули загружаются и компонуются динамически (загрузка по запросу происходит аналогично DLL, однако в отличие от DLL или VBX среды Windows модули Oberon разрешается расширять на этапе выполнения). Для динамического распределения памяти используется системная разделяемая область. При отсутствии ссылок на какой-либо блок он автоматически возвращается "сборщиком мусора" в пул доступной памяти. При наличии подходящего модуля любой редактор документа Oberon способен обрабатывать соответствующий тип представления или реквизита, поэтому само понятие принадлежности файла к тому или иному приложению теряет всякий смысл. Наряду с переменными, определениями и процедурами каждый модуль может содержать так называемые "команды" — специальные, не имеющие параметров экспортируемые процедуры. Такие команды выполняются в любой части системы и вызываются пользователем непосредственно из UI-интерфейса. Чтобы сформировать адрес перехода, система образует пары "модуль.процедура". Команды можно вызывать как явным образом, щелкнув мышью на паре "модуль.процедура" (если это предусмотрено), так и косвенно, в результате взаимодействия с реквизитом. Информация, необходимая для выполнения команды, передается из UI-интерфейса через глобальные переменные (команды могут работать с глобальными переменными как собственных, так и импортированных модулей).

В системе Oberon предусмотрены стандартные модули для ввода с клавиатуры, для реакции на действия мыши, для работы с файлами, управления системой и т. д. Имеется также несколько стандартных абстрактных типов данных (например, текст или растровое изображение) и редакторы для работы с ними. При написании новых модулей вы можете использовать все существующие, причем их исходный код вам не потребуется (компилятор генерирует для каждого модуля расширение интерфейса). Так, при выполнении стандартных команд редактирования расширение редактора будет динамически компоноваться со стандартным модулем редактора.

Сообщения

В модуле Objects определены сообщения, на которые должен отвечать каждый компонент Oberon. Как правило, используются сообщения для записи, копирования или загрузки объекта и сообщения для копирования атрибутов объекта. Их должны воспринимать все компоненты. Обычно объекты отвечают на сообщения (передаваемые через стек) с помощью специальной переменной процедурного типа, определяющей тип воспринимаемых объектом сообщений (на которые он обязан реагировать). По соглашению предусмотрена только одна такая переменная, хотя вы можете определить и дополнительные. Используя расширения типов сообщений, можно наращивать стандартную иерархию объектов и строить дополнительную иерархию типов, создавая, например, собственный объект, тип сообщения и специальную переменную процедурного типа. Характер действия, выполняемого по сообщению, помогает определить явная проверка типа каждого сообщения. Если в C++ вам не нужно инициализировать обрабатывающие методы или определять типы сообщений, то в исходном варианте Oberon это

было обязательным. Однако в Oberon-2 данное требование исчезло (отдельные специалисты полагают, что это привело к некоторой потере гибкости).

Чтобы объекты могли передавать сообщения дальше, не отвечая на них, вам нужно создать в системе Oberon расширения базового типа сообщения, определяющие новый базовый тип. Кроме того, с помощью специального поля сообщения (а само сообщение считается записью) можно определять разную семантику реакции на сообщения одного типа.

Своеобразный механизм обработки сообщений в Oberon отвечает требованиям, предъявляемым к расширяемым системам. В отличие от других объектно-ориентированных языков, где вы должны знать тип объекта, в котором активируете метод (а каждый объект реагирует только на известные ему сообщения), здесь можно передавать сообщения объектам, о которых вы ничего не знаете, а объекты будут передавать не предназначенные им сообщения другим объектам.

Необходимость реакции объектов и на такие сообщения определяется структурой системы и наличием объектов-контейнеров — специальных объектов, управляющих другими (дочерними) объектами. Способность же взаимодействовать с теми объектами, о которых ничего не известно, обеспечивает расширяемость системы. Вам нужно лишь знать, что компонент является объектом, и понимать тип его сообщений. Поскольку воспринимаемые объектом сообщения определяются процедурной переменной, объект реагирует на сообщения, описанные в различных модулях и даже принадлежащие к различным классам (что можно интерпретировать как своего рода множественное наследование для сообщений).

Кроме того, вы можете модифицировать поведение объекта, не изменяя определения типа (что повлияло бы на все расширения данного объекта). Например, разрешается задать реакцию объекта на большее число сообщений (такая реакция нередко составляет проблему, потому что не всегда очевидно, на какие сообщения должен отвечать объект). Каждое сообщение имеет поле результата, в котором объект должен сообщать, ответил он на него или нет.

В Oberon предусмотрен набор сообщений для взаимодействия с доступными в системе текстовыми редакторами и функциями. Это позволяет создать, например, подсистему проверки синтаксиса, работающую со всеми доступными текстовыми редакторами, не прибегая для этого к специальным языкам сценариев.

Программирование приложений

Программировать с помощью Oberon можно на разных уровнях, начиная от простейшего, такого как интерактивное создание документов и интерфейсов и определение для них достаточно простого поведения, до написания фрагментов кода Oberon для работы с реквизитами и программирования своих собственных реквизитов (простых реквизитов или сложных контейнеров, управляющих дочерними реквизитами). Таким образом, под программированием следует понимать как расширение существующего реквизита (подход, знакомый многим приверженцам языков ObjectPAL, Visual Basic и др.), так и создание совершенно нового. В Oberon предусмотрены шаблоны кода, которые можно быстро модифицировать для выполнения требуемых функций. Oberon/F System Как и предшествующие версии системы, Oberon/F реализует высокоэффективную модель составного документа, позволяющую встраивать приложения одно в другое и достигать любого уровня сложности. В проекте Oberon с самого начала одним из ключевых принципов считалась простота, и бета-версия Oberon/F занимает лишь одну 3,5-дюймовую дискету и около 4 Мбайт на жестком диске. Среды программирования, редактирования и отладки здесь полностью интегрированы, и работа с Oberon напоминает такую систему, как Smalltalk. В ее состав входит быстрый компилятор языка Oberon-2, позволяющий получать 32-разрядный код процессоров 486 или 680x0. Этот компилятор работает с высокой скоростью, а компилируемые модули обычно невелики. Oberon имеет удобную среду для написания, компиляции, загрузки и тестирования модулей, основанную на модели составных документов. В окне Show Loaded Modules вы можете подсветить любой модуль в списке и декомпилировать его определение интерфейса. В окне отладки разрешается следовать по указателям и спискам, а ошибки отмечаются в тексте маркерами. Щелчок мышью на таком маркере даст вам соответствующее сообщение об ошибке. Для обеспечения межплатформной переносимости и расширяемости Oberon спроектирован на основе иерархии абстракций. Работа с физическими системами (экраном, печатью, файловой системой) скрыта в абстрактных объектных классах, к которым можно обращаться посредством создания специальных объектов. Например, фундаментальный тип Store представляет данные и соответствующий механизм записи и считывания с диска. Модуль Store реализует "считывающие" и "записывающие" объекты, отображающие типы данных языка Oberon в двоичные данные. Кроме того, Store может представлять составные документы. В первой версии Oberon/F предусмотрены только две компонентные подсистемы - Texts и Forms, что вряд ли можно считать значительным

улучшением по сравнению с предыдущей редакцией. Однако во второй версии планируется реализовать подсистему баз данных в стандарте ODBC (Open Database Connectivity). Существуют также планы включения в Oberon дополнительных компонентов, разрабатываемых прочими фирмами. Поддержка в будущих версиях Oberon OLE 2.0 позволит реализовать "редактирование на месте" стандартных текстовых и графических форматов. Подсистема Texts, как и другие модули Oberon, подлежит расширению. Таким образом, у вас есть возможность дополнить этот текстовый процессор (с функциями, аналогичными Windows Write) нужными вам функциями. Ими можно воспользоваться при работе в системе Oberon с любым текстом, а модуль Texts при этом даже не потребуется перекомпилировать. Полученный в результате код будет функционировать сходным образом на различных платформах (пока это Windows и Macintosh). Компонент Forms представляет собой простое средство визуального проектирования форм ввода данных и диалоговых окон. После написания кода, определяющего тип данных различных полей, вы проектируете форму, буксируя в нее объекты управления (аналогично Visual Basic). Исполняющая система Oberon/F создает и обслуживает связи между полями экрана и соответствующей структурой данных. При вводе данных в какое- либо поле автоматически обновляется описанная для него переменная. Если же поле обновляет программа, то она посредством сообщений должна оповестить об этом определенные экранные представления и реквизиты. Формы Oberon/F также хранятся в виде документов, но в отличие от обычных генераторов кода вы можете изменить их вид, не перекомпилируя приложение. Формы можно встраивать в текст, получая сложный составной документ и реализуя разнообразные интерфейсы с пользователем.

Заключение

В стремлении производителей не отстать от всех технологических новшеств в программных разработках нередко теряется основная цель — сделать работу пользователей более простой и продуктивной. Концепции новинок часто настолько сложны, что требуют немалых усилий для их освоения и изучения. Это, естественно, многих отпугивает, и переход на новую технологию обычно связан с большими проблемами. Все это в полной мере относится и к объектной технологии. Хотя можно найти примеры успешной реализации объектно-ориентированных методов, распространяются и используются они отнюдь не так быстро, как можно было бы ожидать. Известные слова о том, что дело не в технологии, а в людях, никогда не были столь справедливы, как в данном случае. "Человеческий фактор" имеет здесь решающее значение. Объектные принципы требуют совершенно новых подходов, одним из которых является компонентное ПО. В конечном счете оно позволит разбить монолитные приложения на компоненты, а пользователи получат возможность персонализировать свои приложения, составляя их из готовых фрагментов.

Это способно совершенно изменить общепринятую стратегию создания программного обеспечения. Вместо приложений можно будет распространять компоненты и фрагменты, ориентированные на решение конкретной задачи. Компонентное ПО позволит получать комплекты приложений путем простой сборки фрагментов. Такие комплекты будут собираться по заказу или поставляться в виде стандартных решений для конкретных задач. Технологии компонентного ПО фактически способны превратить рынок программных средств в рынок компонентных модулей. Компании-разработчики могут создавать и продавать программные модули с соответствующими стандартными интерфейсами, из которых пользователи (или другие компании) будут собирать ПО с нужными функциями. В области программного обеспечения такие концепции, как "компонентное программное обеспечение" и "взаимодействующие объекты", становятся все более популярными и скоро, видимо, станут атрибутами современных систем. Oberon — весьма интересная и перспективная разработка, свидетельствующая о том, что с помощью объектно-ориентированного программирования можно создавать расширяемые модульные системы, и переход к действительно компонентному ПО, по-видимому, уже не за горами. Oberon является одним из первых практических шагов в этом направлении.