

цедуры графики и пользовательского интерфейса были четко выделены и легко встраивались в наши программы сбора данных. И программа, которая изначально была написана, как демонстрация пакета (MMSLPLT), стала чем-то вроде стандарта в нашей лаборатории.

После этого мы реализовали ряд программ, обеспечивающих сбор данных и пропорциональное управление для мультизоновых печей, прецизионный контроль температуры ($\pm 0.002^\circ\text{C}$) для экспериментов по низкотемпературному росту кристаллов, создали мультикомпьютерную сеть со стендом для телевербатики, реализовали систему сбора данных и управления установкой плазменного напыления. К тому же мы знаем программиста, который написал очень сложный диспетчер для управления целым комплексом тестовых систем.

Наш опыт работы с Modula-2 был не очень велик, но крайне положителен. Мы обнаружили, что этот язык подходит для реализации практических всех наших проектов. И все же нам хотелось бы увидеть некоторые дополнительные средства (вроде операции возведения в степень!) и побольше разных библиотек. Если говорить в общем, то нам язык очень понравился. Мы следим за новыми разработками (вроде Modula-3 и Oberon-2), но уходить от Modula-2 пока не собираемся.

Микаэль Франц

Проект OMI и кодогенерация на лету

Michael Franz (1994) «The Oberon Module Interchange (OMI) // Oberon News, No.3, p.2-4.
© 1994, Michael Franz

Микаэль Франц (1995) «Проект OMI и кодогенерация на лету // Технология программирования, Т.1, №1, с.183.
© 1995, Руслан Богатырев, перевод с англ.

«Эта программа для Macintosh, она не пойдет на мой PC». Прошло вот уже 15 лет с момента появления персонального компьютера, и теперь даже профаны в области техники понимают, что существуют разные виды компьютеров, которые требуют различного программного обеспечения. Хотя это разнообразие приводит к высокой стоимости разработки и распространения программ, эту ситуацию воспринимают как непреложный факт нашей жизни.

Проект OMI (Oberon Module Interchange) неставил своей целью ничего иного, кроме как изменить этот устоявшийся взгляд. Вводится концепция «переносимого объектного файла», который можно использовать более чем на одной разновидности целевой машины. Механизм OMI полностью проясчен для пользователя: не нужно использовать никаких конвертеров или инсталляционных процедур, чтобы мож-

но было использовать конкретный объектный файл на компьютере с поддержкой OMI. Единственное различие между переносимыми и обычными объектными файлами с точки зрения пользователя состоит в том, что первые можно использовать на более широком диапазоне платформ, нежели вторые.

*Все это хорошо.
Но как оно работает?*

Должно быть очевидно, что переносимые объектные файлы не могут содержать реального объектного кода. Вместо этого они содержат абстрактное представление программы с применением специального кодирования, которое в дальнейшем позволяет осуществлять быструю генерацию соответствующего чистого кода для конкретной целевой машины. Причем эта генерация производится во время загрузки программы. Поскольку механизм OMI встроены в систему Oberon, в которой модули могут подгружаться динамически в любой момент во время сеанса работы, это означает, что OMI может последовательно компилировать одни части программной системы, когда другие уже работают. Например, когда пользователь (или программа) вызывает команду из переносимого модуля, который еще не загружен, OMI просыпается, компилирует этот модуль на лету (on-the-fly), выполняет только что полученное тело модуля и возвращает управление обработчику команд, который затем переходит на новую команду.

Кодогенерация на лету во время загрузки была бесполезной, если бы она работала слишком медленно (что неудобно для интерактивного пользователя) и если бы порождала плохой код. К счастью, обе эти проблемы решены в механизме OMI. В своей первой реализации OMI мог компилировать и загружать переносимые объектные файлы примерно за то же время, которое требовалось для считывания с диска обычных объектных файлов. Причиной этого состоит в том, что переносимые объектные файлы гораздо меньше, чем их настоящие собратья. А время загрузки программы в основном зависит от быстродействия ввода/вывода. Поскольку производительность процессоров растет гораздо быстрее быстродействия устройств хранения информации, очень даже вероятно, что кодогенерация на лету превзойдет загрузку традиционных объектных файлов.

*В теории все звучит хорошо.
А как на практике?*

OMI — это исследовательский проект Института компьютерных систем, входящего в состав Швейцарского федерального технологического института ETH в Цюрихе. Он представляет собой докторскую работу автора. В настоящее время OMI доступен только на платформе Macintosh (MC680x0), но в ближайшее время начнут поддерживать и другие платформы. Существует несколько причин выпуска OMI уже сейчас, еще до того, как другие реализации сделают его гораздо полезнее. Прежде всего, я надеюсь, что существующая реализация убедит даже са-

мых завзятых скептиков. Во-вторых, OMI представит внешнему миру идею, где должны появиться будущие версии системы Oberon. В-третьих, существование OMI в значительной степени способствует росту интереса к Oberon'у и обнародование результатов в начальной стадии может предотвратить некоторые неверные решения в отношении приоритетов реализации. И, наконец, невозможно откладывать сложную систему без обратной связи с ее пользователями. Многие уже экспериментируют с OMI, дают свои замечания и представляют отчеты об ошибках, так что OMI к моменту появления других реализаций уже выйдет на промышленный уровень.

Важно заметить, что в начале 1995 года появится реализация OMI для Apple PowerMacintosh, которая сформирует ядро нового релиза Mac-Oberon, работающего в native-режиме и на процессорах MC680x0, и на процесорах PowerPC.

До сих пор проблема запуска программ более чем на одном виде компьютеров решалась с помощью подхода, получившего название «толстые бинарники» (fat binaries). При этом фактически несколько исполняемых версий программы находятся в одном «файле прикладной программы», по одной версии для каждого типа центрального процессора, и система в момент запуска выбирает, какую из частей ей использовать. Конечно использование толстых бинарников приводит к увеличению размеров программ в несколько раз по сравнению с их первоначальными габаритами. Обладая OMI-технологией, будущий MacOberon сможет поддерживать работу с «тонкими бинарными» (slim binaries). Программисты смогут выпускать свои модули для использования всем сообществом OMI, при этом не будет необходимости раскрывать исходные тексты. Это еще на один шаг приблизит нас к старой мечте «программных компонент», что позволит революционно изменить ситуацию в дисциплине программной инженерии.