Юрг Гуткнехт

Oberon в системе образования

Jurg Gutknecht (1995) Oberon in Education // The ModulaTor, 1995, Vol.5, No.6.
Р. Богатырев, А. Китаев, перевод с англ.

Настоящая работа представляет собой тезисы выступления профессора Ю. Гуткнехта на Дне Oberon (the Oberon Day 1994) в Швейцарском федеральном технологическом институте (ETH, Цюрих). Кратко рассмотрены три основные концепции программирования: модуляризация, абстрактные типы данных и классы, а также особенности операционной системы Oberon.

Проектирование программных систем столь же разительно отличается от обычного программирования, сколь отличается строительство архитектурных сооружений от производства кирпичей. В первом случае мы берем простые элементы (кирпичи) и создаем с их помощью сложные структуры (здания); тогда это называется программированием-в-большом (programming in the large).

Название "Oberon" относится одновременно и к (операционной) системе, и к языку программирования [1, 2], и в этом отражается идея цельности и интегрированности.

Oberon, вызревавший в ходе длившейся несколько десятилетий эволюции [3], дает нам многое в понимании того, как следует работать с большими проектами. Очень важно уже то, как возникла конструкция языка программирования. Oberon можно преподавать в том виде, в каком он существует, ибо эта конструкция прошла через все стадии программной инженерии. В отличие от других объектно-ориентированных языков, например, C++, который представляет собой своеобразный плавильный тигель со множеством различных концепций низкого и высокого уровня, Oberon, во-первых, четко разделяет различные концепции, и, во-вторых, унифицирует похожие и одинаковые конструкции.

По своей сути C++ — это комбинация языка Си (который ненамного сложнее обычного языка ассемблера) и конгломерата различных концепций (таких, как функции, файлы, типы, классы, наследование, повторное использование и т.д.). Именно поэтому на примере C++ трудно изучать отдельные концепции. Образно говоря, C++ — это не одно, а несколько блюд, поданных под одним соусом.

Мы можем грубо выделить три следующие концепции программирования.

1. Модуляризация (Modularization)

Модуляризация — это важная концепция программирования-в-большом. Все иерархии модулей являются статическими, хотя и могут расширяться. Статическая структура лучше обеспечивает проверку целостности как во время трансляции, так и во время выполнения программы и дает неплохую наглядность и цельное восприятие. Хорошим примером нетривиальной модульной структуры служит сама система Oberon.

С появлением модульных систем исчезает разрыв между системными и прикладными программистами. И те, и другие организуют импорт из интерфейсов абстрактных модулей. В системе Oberon функциональная модуляризация реализована в текстовой подсистеме [2], которая ответственна за обработку текстов, шрифтов и представлений дисплея.

2. Абстрактные типы данных (Abstract Data Types, ADT)

Абстрактные типы данных обеспечивают абстрактное представление объектов, а также полный набор соответствующих операций. В системе Oberon примером ADT могут служить файлы [2]. Модуль Files экспортирует абстрактный тип данных File вместе со всеми необходимыми операциями.

3. Проектирование классов

Проектирование классов используется для создания специализированных цепочек, представляющих типы объектов. Обычно такие цепочки короткие, как правило, они состоят из двух уровней. Первый уровень ("виртуальный") характеризуется высокой степенью абстракции, и каждая конкретная реализация представляет его специфику. Полиморфизм [4] во время выполнения позволяет динамически выбирать соответствующие операции.

Если модули обычно содержат иерархию функций (функциональность), которые можно импортировать, то классы устанавливают основу наследования. Нужно особо подчеркнуть, что наследование не равнозначно импорту: понятия многократно используемой функциональности и полиморфического поведения весьма отличаются друг от друга. В обычных объектно-ориентированных языках существует только концепция класса и нет иерархии модулей, так что четкое разделение между двумя концепциями (импорт и наследование), к сожалению, отсутствует. Типичным представителем такой модели является язык Smalltalk.

В Oberon концепции типов и классов объединены. Это логично, поскольку классы и типы можно рассматривать просто как два различных представления одной и той же конструкции. Тем не менее, некоторые языки, как, например, C++, оперируют обеими концепциями — как типами, так и классами, а это не только не облегчает работу, но даже осложняет ее.

Класс в Oberon'е представлен как тип записи. Механизм расширения типа (type extension) представляет собой специализированный инструмент для описания подклассов. Стоит отметить, что расширение типа используется в Oberon и для создания структур данных — как родовых (generic), так и неоднородных (heterogeneous). Так, неоднородный список состоит из элементов, строгая идентифицированность которых может изменяться. Хорошим примером неоднородной структуры данных в системе Oberon может служить так называемый ориентированный ациклический граф (Direct Acyclic Graph, DAG) — ориентированный граф пространства отображения, управляемый модулями Frames и Gadgets [5].

Родовые интерфейсы сообщений — еще одна интересная конструкция Oberon. При работе с такими интерфейсами проектировщикам программного обеспечения необязательно раз и навсегда закреплять за данным классом определенный набор сообщений. Родовые интерфейсы сообщений позволяют объектам обрабатывать "неизвестные" сообщения или, в крайнем случае, принимать такие сообщения и передавать их потомкам дальше по цепочке. А это важно для таких, например, сообщений, которые передаются в пространство отображения. Концепция типа допускает к тому же дальнейшее расширение протокола сообщений. Процедурные переменные применяются в Oberon для представления методов в классах (типах записи) и для так называемых отложенных вызовов (up-calls). Отложенные вызовы обычно совершаются операционной системой по требованию. Хорошей сферой приложения отложенных вызовов являются родовые драйверы устройств, т.е. драйверы абстрактных устройств, которые вызывают установленные функции.

Абстрактные типы данных в системе Oberon можно рассмотреть на примере модуля Texts. Для систем редактирования текстов характерна сложная реализация. Однако эта реализация скрыта абстрактным интерфейсом, не позволяющим клиентам ничего знать о деталях реализации, с тем чтобы ничего не испортить во внутренней организации системы. Подобная инкапсуляция дает еще одну интересную возможность — свободно изменять или модифицировать реализацию, сохраняя при этом функциональность клиентных модулей. А это, разумеется, немаловажное обстоятельство при работе над крупными проектами.

Операционная система Oberon весьма многогранна. Она включает в себя:

- разносторонний пользовательский интерфейс:
- мощный интерфейс программирования;
- богатую модульную архитектуру;
- набор инструментальных средств (компилятор, редактор и т.п.);
- основу для программирования-в-большом.

Программирование на языке Oberon эквивалентно расширению иерархии модулей. Модули на вершине иерархии можно рассматривать как прикладные пакеты инструментальных средств, а модули нижних уровней как системно-ориентированные. Внутреннее содержание модулей, а также требования к их устойчивости (к нежелательным воздействиям) и корректности могут быть различными, в зависимости от их позиции в иерархии модулей, но их форма и структура одинакова на всех уровнях. Так, качество модулей, находящихся на нижних уровнях, должно быть исключительно высоким, поскольку их нельзя изменить без перезапуска системы.

Благодаря тщательно продуманной конструкции своих модулей, система Oberon представляет собой хороший материал для изучения программирования-в-большом, включая нетривиальные примеры абстрактных типов данных и классов. Такие примеры имеют исключительную ценность при преподавании и изучении концепций профессионального программирования. Еще одним важным аспектом преподавания является наглядность и глубокое понимание системы, о которой идет речь.

Аппаратная база постоянно развивается, и все ждут теперь нового шага вперед в разработке программного обеспечения, скажем, в сторону создания сохраняемых (persistent) и визуальных объектов. Соответствующим примером в Oberon System 3 может служить система GUI-приспособолений (gadget) [5]. Приспособления поддерживают интерактивную и программируемую конструкцию сохраняемых объектов. Gadget-программистам не нужно изучать новый язык. Однако тем из них, кто захочет использовать весь потенциал приспособлений и, в частности, предоставляемую ими возможность программировать настраиваемые приспособления, потребуется знание общесистемного протокола сообщений, состоящего из 6 базовых сообщений и 7 дополнительных сообщений для визуальных объектов. При создании новых приспособлений можно воспользоваться скелетными модулями в исходной форме. Это очень удобно, ибо, вообще говоря, модификация представляет собой гораздо более простую операцию, нежели разработка (программы) "с нуля".

В Oberon System 3 иерархия объектов содержится в ядре системы, в классе Objects.Object. Важно отметить, что благодаря уже упоминавшемуся разделению импорта и наследования, иерархия объектов в Oberon не характеризуется большой вложенностью и состоит, как правило, из трех уровней. С другой стороны, концепции автономных объектов (без их иерархической организации) оказалось недостаточно.

Для создания наборов объектов требуется организующее инструментальное средство. В Oberon System 3 такие наборы называются библиотеками. В своей совокупности библиотеки объектов составляют иерархию, которая дополняет иерархию модулей. Если последняя применяется для структурирования функциональности системы, то первая организует устойчивые компоненты системы.

Литература

[1] Wirth N., Gutknecht J. (1992) Project Oberon: The Design of an Operating System and Compiler // ACM Press.

[2] Reiser M. (1992) The Oberon System: User Guide and Programmer's Manual // ACM Press.

[3] Wirth N. (1988) From Modula to Oberon // Software: Practice and Experience, Vol.18, No.7, p.661-670.

[4] Moessenboeck H. (1993) Object-Oriented Programming in Oberon-2 // Springer-Verlag.

[5] Marais H. (1993) Gadgets: Principles and Programming // ETH Zurich.

[6] Reiser M., Wirth N. (1992) Programming in Oberon: Steps Beyond Pascal and Modula-2 // ACM Press.

[7] Moessenboeck H., Wirth N. (1991) The Programming Language Oberon-2 // Structured Programming, Vol.12, p.179-195.

[8] Wirth N., Gutknecht J. (1989) The Oberon System // Software: Practice and Experience, Vol.19, No.9, p.857-893.

[9] Wirth N. (1988) The Programming Language Oberon // Software: Practice and Experience, Vol.18, No.7, p.671-690.

Об авторе. Юрг Гуткнехт (Juerg Gutknecht) — известный педагог, профессор Швейцарского федерального технологического института, принимал активное участие в проекте Oberon. По сути является соавтором Н. Вирта в создании языка Oberon, сыграл большую роль в доведении основополагающей концепции этого языка, концепции расширения типа (type extension), до математической основы в виде проекции пространств. Большой популяризатор языка и системы Oberon. Сферой его профессиональных интересов являются операционные системы и системы пользовательского интерфейса.