Руслан Богатырев

От Паскаля к языку Zonnon: реализация новых идей на платформе .NET

Источник: Мир ПК, #09/2003.

Мир современных языков программирования развивается под знаком яростного противостояния Java и C#. И все же новые языки, возрождающие славные традиции прошлого, не боятся бросить вызов признанным лидерам.

Куда мы идем?

Языки программирования, несмотря на все свое бесконечное разнообразие и значительную адаптацию к особенностям компьютерной обработки, еще не утеряли связи с привычными нам естественными языками, что в общем-то объяснимо, ведь писать на них должны люди, а не машины.

Как и в естественном языке, в программировании мы встречаемся с повелительными конструкциями (императивные языки, такие как Фортран, Си и Паскаль), изъявительными (декларативные языки, например Лисп и Пролог) и вопросительными (языки запросов, в частности SQL). В известной степени их всех объединяет повествовательная форма (сценарные языки, подобные Perl, Python), а также текстуальное структурирование (языки разметки вроде HTML- и XML-семейств).

Перечисленные группы языков значительно различаются синтаксисом и семантикой. Более того, можно даже утверждать, что пишущие на них программисты должны мыслить по-разному. Не думаю, что здесь стоит погружаться в глубины психо- и нейролингвистики или выяснять, насколько сильно при использовании таких языков различается работа левого и правого полушарий мозга. Но даже на интуитивном уровне понятно, что конкретный язык программирования, подобно естественному языку, в значительной степени диктует способ мышления.

Различия наблюдаются не только между группами языков, но и внутри них. Наверное, не будет преувеличением сказать, что среди императивных языков с точки зрения синтаксиса можно выделить три доминирующих направления: Фортран-ориентированное (Фортран, Кобол, Visual Basic), Паскаль-ориентированное (Паскаль, Delphi/Object Pascal, Ada), Си-ориентированное (Си, C++, Java, C#). Конечно же я намеренно упрощаю реальную картину, но при изложении новых идей лучше отталкиваться от понятных эталонов.

Фортран-ориентированное направление привлекает к себе людей, связанных с прикладными задачами программирования, т. е. тех, кто посредством языка решает весьма специфические задачи, будь то из области математики или бизнеса. Паскаль-ориентированное направление чаще выбирают люди, следующие классическим канонам программирования и четким принципам, заложенным еще на университетской скамье (программирование как наука и искусство). Наконец, Си-ориентированное направление стало уже общепризнанной нормой промышленного производства программного обеспечения, другими словами, неотъемлемой частью программирования как ремесла. К сожалению, приходится признать, что мир профессионального программирования, выбрав языки Си-семейства, давно уже вступил на скользкий путь избыточной сложности.

В результате такие критерии, как простота (наглядность исходных текстов), надежность, компактность программ и эффективность исполняемого кода, теперь заменяются на один главный критерий — скорость реализации идей. А это неизбежно ведет к резкому снижению влияния самого языка и возрастанию роли инструментальной среды, которой подчас вообще все равно, чем оперировать, и прежде всего ее возможностей визуализации. Как точно отмечает Никлаус Вирт, «постоянный недостаток времени — вот, вероятно, первейшая причина, приводящая к появлению громоздкого программного обеспечения» [1]. Приоритет времени разработки в общем-то понятен: человеческие ресурсы нынче стоят значительно выше, чем аппаратные. Но не теряем ли мы по дороге то ценное, что вернуть потом будет крайне тяжело?

Лидеры компьютерной индустрии, следуя законам маркетинговых войн, последовательно внедряют в массы все новые и новые мифы (что именно компиляторы C++ формируют наиболее эффективный код, что именно Java создает самые переносимые программы и что именно C# отражает самые передовые концепции программирования).

Сейчас редко кто ставит под сомнение целесообразность почти повсеместного использования доминирующей объектно-ориентированной парадигмы. Не замечая этого, мы создаем собственные химеры и лишь со временем — как здесь не вспомнить бессмертную сказку Андерсена — прозреваем и понимаем, что король-то голый.

Важное достоинство .NET

Какой бы критике мы ни подвергали нынешние промышленные языки (что имеем, то имеем), справедливости ради стоит сказать, что в результате их конкуренции все же сложились благоприятные технологические условия. Для чего? Чтобы дать возможность другим языкам (новым и старым) стать полноценными участниками созидательного процесса программного строительства и не остаться на обочине ИТ-прогресса. Конечно, хорошо жить в созданном тобою же мирке, тихом и уютном, не подверженном бурям и переменам клокочущего мира, а не покорно плыть в бушующем море, полностью отдавшись во власть штормам и течениям. Но как это сделать, если только не отгородиться от этого безумного мира стеной таких абстракций, которые было бы несложно соединять коммуникациями со все новыми и новыми продуктами и технологиями программной индустрии?

Давайте порассуждаем. Допустим, вы придумали новый язык и смогли формализовать его конструкции. Но это даже не полдела. Чтобы начать писать на нем программы, надо пройти огромную дистанцию от создания транслятора (компилятора или интерпретатора) до реализации вспомогательных инструментальных средств. Такой путь займет не один год, и к моменту завершения своего ратного подвига вы увидите, как далеко за это время ушла индустрия, не оставив вам шанса создавать на новом языке практические и полезные программы. Вы останетесь один на один со своим детищем, продолжая вдыхать в него жизнь, бережно прижимая его к сердцу и сокрушаясь о том, как несправедлив окружающий мир.

А теперь поставьте себя на место потенциальных пользователей этого языка: они тысячу раз подумают, прежде чем тратить свое время и связываться с языком, о котором еще не написано достаточного количества книг и для изучения которого не составлено пособий, хороших программных библиотек, качественных компиляторов и отладчиков. И все же...

Появление языковой платформы CLI (Common Language Infrastructure) для .NET (как тут лишний раз не сказать доброе слово тем профессорам, что помогали Microsoft Research и большому Microsoft создавать эту альтернативу моноязыковой Java-платформе) позволило совершить главный прорыв: унифицировать межъязыковое взаимодействие и значительно облегчить процесс внедрения нового языка в единую инструментальную среду. Теоретически теперь достаточно лишь придумать язык либо позаимствовать какой-то полузабытый и написать для него компилятор переднего плана (front-end compiler), формирующий промежуточный код, а вот генератор исполняемого кода можно будет использовать штатный, из .NET.

Также (опять-таки теоретически) можно будет относительно просто встроить свой транслятор в среду Visual Studio .NET, после чего останется лишь наслаждаться своим произведением. Благодаря межъязыковой унификации можно будет легко и непринужденно использовать библиотеки, созданные на других языках, да и вообще задействовать свой язык только на самых критических участках, где он даст фору всем остальным (а иначе зачем его было создавать и реализовывать?). В общем, рай, да и только.

Понятное дело, что в действительности на этом пути к всеобщему благоденствию ждет немало неприятных сюрпризов. Тем более интересно узнать обо всех подробностях из уст первопроходцев, решивших во благо будущего программирования проложить путь по еще не хоженым маршрутам. К тому же они были первыми (за пределами Microsoft), кто испытал на себе возможности нового инструментария CCI (Common Compiler Infrastructure), созданного в Microsoft.

Где и как возник Zonnon

Язык Zonnon является продолжателем принципов, заложенных в языках Паскаль (1970), Modula-2 (1979), Oberon (1988). Все эти языки объединяет одно: они были созданы известным ученым Никлаусом Виртом (Niklaus Wirth) в ведущем европейском университетском центре ETH Zurich (Swiss Federal Institute of Technology). Язык Zonnon в этом смысле не исключение: он проектировался в том же знаменитом центре Юргом Гуткнехтом (Jurg Gutknecht), многолетним соратником Вирта по проектам Lilith/Modula-2 и Ceres/Oberon [2].

Не знаю, стоит ли читателю представлять профессора Вирта, который в будущем году будет отмечать свое 70-летие. Приведу лишь один малоизвестный факт. Еще в 1982 г. Рональд Рейган написал письмо, адресованное 20 ученым и инженерам в области компьютерных систем. «Я очень рад передать свои самые сердечные поздравления первопроходцам, — начал его президент США, — чьи имена навечно вписаны в Зал Славы компьютерной индустрии». Тот список охватил практически всех известных деятелей компьютерной промышленности от Томаса Уотсона, патриарха IBM, и Билла Гейтса, главы Microsoft, до Стива Джобса, основоположника Apple. И хотя мировому компьютерному сообществу были хорошо известны эти новаторы и изобретатели, все же один из лауреатов выпадал из общего ряда. Никлаус Вирт не имел ни одного патента в области электроники, никогда не возглавлял ни одну из известных высокотехнологичных компаний и даже не являлся гражданином США. Его путь в Зал Славы лежал через скромные научные отчеты. На протяжении десятков лет он закладывал основы и самостоятельно реализовывал семь языков программирования (PL360, Algol-W, Паскаль, Modula, Modula-2, Oberon, Oberon-2). Именно их создание и привело к тому, что швейцарскому профессору дали имя «отца структурных языков».

Самая критически важная составляющая программных систем — надежность. Она напрямую связана с простотой и лаконичностью. Вирт отмечал: «Поддержание языка максимально простым и регулярным всегда было приоритетом в моей работе: описание Паскаля занимало около 50 страниц, Modula-2 — около 40, а Oberon — и вовсе 16. И я рассматриваю эту тенденцию как прогрессивную. Истинная ценность языков программирования зависит от качества и практичности их абстракций». Интересно отметить, что предварительный вариант описания языка Zonnon [3], подготовленный Ю. Гуткнехтом и Е. Зуевым (известным экспертом по компиляторам языков Ada и Си++, защитившим диссертацию в МГУ, а ныне старшим научным сотрудником ETH Zurich), содержит около 30 страниц (при этом включает достаточно большие примеры исходных текстов).

Стоит отметить, что Ю. Гуткнехт и Е. Зуев принимали активное участие в сотрудничестве с Microsoft Research в рамках проекта Project 7/7+ (разработка компиляторов для CLR, Common Language Runtime) в период проектирования и реализации платформы Microsoft .NET (1999—2002). Разумеется, в ходе разработки (по предложению Microsoft) компилятора Oberon for .NET и постоянных контактов с сотрудниками Microsoft Research (среди которых есть и выпускники ETH) ими был накоплен достаточно богатый опыт работы с новой языковой платформой.

Основные особенности

Ю. Гуткнехт, как руководитель и идейный вдохновитель проекта Zonnon, постарался воплотить в новом языке те ключевые принципы, которые заложил Вирт. При этом, проанализировав опыт Java и C#, он заложил в Zonnon новую концептуальную модель, базирующуюся на идее активных объектов, опробованной им в языке Active Oberon. Но самое главное, он сумел построить органичную интеграцию модульного и объектного программирования с помощью четверки ключевых понятий: Object (объектный тип) — Definition (описание) — Implementation (реализация) — Module (модуль).

Проект Zonnon призван показать, что для .NET можно эффективно реализовывать универсальные императивные языки, в некоторых аспектах превосходящие C# и исповедующие принципиально иную концептуальную модель.

Чтобы ее лучше понять, имеет смысл вернуться к истокам. В Паскале Вирт блестяще реализовал фундамент структурного программирования. В языке Modula-2, вобравшем в себя ряд идей из Mesa (язык, созданный в Xerox PARC), в основу уже были положены принципы абстрактных типов данных, модульного программирования и раздельной компиляции описательной (DEFINITION) и исполнительной (IMPLEMENTATION) частей модуля.

Что же касается языка Oberon, то он уже базировался на принципах расширяемости систем. Вирт, значительно подсократив Modula-2 (кое-где, на мой взгляд, напрасно), сумел введением всего одной новой конструкции добиться элегантного воплощения идей объектно-ориентированного программирования. Он ввел понятие расширения типа (type extension) — по сути наследования типов, созданных с помощью конструктора RECORD (запись). А поскольку в качестве типов полей этого конструктора могли выступать и процедурные типы (в частности, затем в C# это было реализовано в виде так называемых делегатов), то класс и его экземпляр (объект) отобразились в Oberon как расширяемый тип и его переменная. Вот и все. Эффективность этого лаконичного языка была доказана на практике, в том числе и Ю. Гуткнехтом, ставшим главным архитектором крайне интересной операционной системы Oberon, способной работать на «голой» машине, на специально для нее спроектированном оборудовании (Ceres) и поверх любой другой ОС (Windows, Mac OS, UNIX). Из нее годы спустя черпали идеи лидеры компьютерной индустрии.

Если сравнивать Zonnon с C#, то главные отличия заключаются в четырех уровнях, выделенных Ю. Гуткнехтом:

- параллельность (Concurrency);
- композиционность (Composability);
- расширяемость (Extensibility);
- синтаксис (Syntax).

Причем каждый из этих уровней наследует идеи конкретного языка виртовского семейства (соответственно, Active Oberon, Modula-2, Oberon, Паскаль).

Параллельность базируется на концепции активных объектов — каналов и процессов, встроенных в традиционные объекты. Название «объект», ставшее эталоном в ООП, вводит в заблуждение, поскольку объекты вовсе таковыми не являются, ведь обычно не они осуществляют воздействие. Скорее это субъекты — на них оказывается воздействие, и они просто реагируют на внешние раздражители, преобразуя сообщения в вызовы методов. Принципы параллельности (на основе асинхронной конструкции AWAIT) и активности объектов были воплощены Ю. Гуткнехтом в языке Active Oberon и затем перенесены с минимальными изменениями в Zonnon. Накоплен богатый практический опыт реализации подобной модели в системах AOS и Bluebottle, созданных в ETH.

Подход к параллельности иллюстрирует небольшой пример взаимодействия двух станций (Station), обменивающихся информацией через порты (буфер объектов), который приведен на рисунке и в листинге 1.

Листинг 1. Объектный тип Station.

```
OBJECT Station (next: Station);

  VAR {PRIVATE}
    in, out: INTEGER;
    buf: ARRAY N OF OBJECT;

  PROCEDURE {PRIVATE}  Get (VAR x: OBJECT);
  BEGIN {LOCKED}
    AWAIT (in # out);
    x := buf[out];
    out := (out + 1) MOD N
  END  Get;

  PROCEDURE {PUBLIC} Put (x: OBJECT);
  BEGIN {LOCKED}
    AWAIT (in + 1) MOD N # out;
    buf[in] := x;
    in := (in + 1) MOD N
  END Put;
```

```
ACTIVITY;
    VAR x: OBJECT;
  BEGIN
    LOOP
      Get(x); (* далее должна идти обработка объекта x *)
      next.Put(x)
    END
  END

BEGIN
  in := 0; out := 0
END  Station.
```

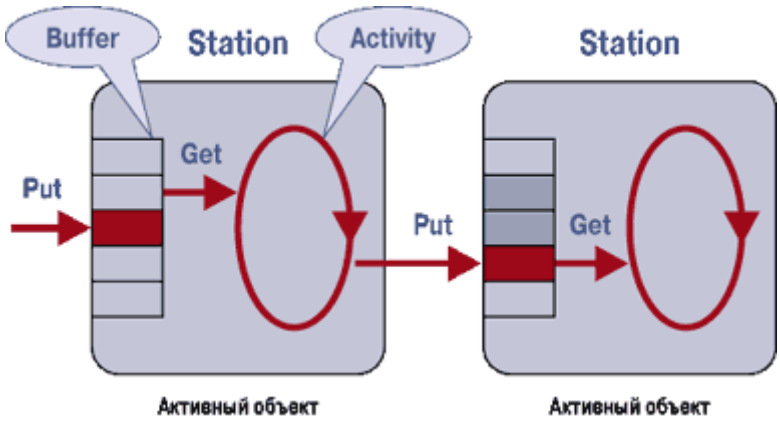

Рис. 1. Схема взаимодействия двух рабочих станций.

Композиционность является, пожалуй, наиболее интересной новацией языка, поэтому о ней расскажем подробнее чуть позже (во второй части статьи). Здесь же отметим, что Ю. Гуткнехт вернулся к истокам — языкам Mesa и Modula-2, дабы вместо весьма мудреной взаимосвязи понятий обычных и концевых (sealed) классов, объектов, интерфейсов, пакетов и пространства имен предложить более четкую структуру. Это позволяет не только получить возможность поддерживать одинарное и множественное наследование, полиморфизм, уточнение и агрегирование, делегирование на уровне сигнатур методов, но и в полной степени реализовать достоинства модульного и компонентного программирования.

Расширяемость в Zonnon продолжает традиции Oberon в направлении родового, или обобщенного, программирования (generic programming). Теперь появляется возможность осуществлять перегрузку методов и операций, вводя собственный синтаксис, чтобы строить собственные протоколы взаимодействия. Кроме того, модификаторы (modifier), дающие возможность изменять семантику программных конструкций, обеспечивают тонкое взаимодействие программиста с компилятором и средой исполнения (своего рода директива/прагмы компилятора и параметры периода исполнения). В приведенном выше примере модификаторами являются PUBLIC, PRIVATE, LOCKED.

Наконец, **синтаксис** Zonnon в максимальной мере использует достоинства Паскаль-ориентированного направления над Си-ориентированным. Причем для тех, кто знаком с классическим Паскалем и такими его диалектами, как TurboPascal и Object Pascal (ныне переименованный в Delphi), стоит подчеркнуть, что раздражающая, ненужная избыточность синтаксических скобок begin-end в Zonnon отсутствует (что Вирт сделал еще в Modula-2 и Oberon), все зарезервированные идентификаторы записываются заглавными буквами (что дисциплинирует программистов и задает четкий визуальный каркас программ), а многочисленные служебные операторы и вариативные конструкции заменены унифицированными модификаторами.

В качестве ремарки в конце первой части статьи отмечу, что в языке есть такие средства, как открытые, квазидинамические и динамические массивы, обобщенные типы (на основе совместимости с ARRAY OF BYTE), универсальные процедуры ввода-вывода (возвращенные из Паскаля), обработка исключений. Убран конструктор RECORD (теперь это просто OBJECT с модификатором VALUE). Ну и, разумеется, никакого оператора goto в виртовских языках нет и в помине. Автоматическая сборка мусора сохранена (как и в Oberon).

Трудно сказать, присутствуем ли мы с вами при рождении «сверхновой звезды», но язык Zonnon, безусловно, заслуживает внимания, причем не только тех, кто привык иметь дело с диалектами Паскаля, но и тех, кто связал свою профессиональную деятельность с C++, Java и C#.На момент написания этих строк готовилась к выходу первая версия компилятора Zonnon, поэтому читателям

можно порекомендовать ознакомиться с реализациями идей в предшественнике Zonnon — языке Active Oberon, а также в системах AOS и Bluebottle.

Автор благодарит Ю. Гуткнехта и Е. Зуева за обстоятельные консультации по особенностям языка и компилятора Zonnon.

Продолжение следует.

Литература

1. Вирт Н. Долой "жирные" программы // Открытые системы. 1996. № 6.
2. Богатырев Р. Летопись языков. Паскаль // Мир ПК. 2001. № 4.
3. Gutknecht J., Zueff E. Zonnon Language Report. Draft // ETH Zurich, June 2003.

Web-ссылки

- Zonnon http://www.bluebottle.ethz.ch/Zonnon/
- Oberon http://www.oberon.ethz.ch/
- Modula-2 http://www.modula2.org/
- Bluebottle http://www.bluebottle.ethz.ch/
- Никлаус Вирт http://www.cs.inf.ethz.ch/ ~wirth/
- Юрг Гуткнехт http://www.cs.inf.ethz.ch/ ~gutknech/
- Евгений Зуев http://www.cs.inf.ethz.ch/~zueff/
- CCI http://www.msdhaa.net/CCI/