Руслан Богатырев

Java на марше

Источник: Мир ПК, #09/2002

С момента фактического рождения языка Java прошло-то всего ничего, каких-нибудь семь лет, а кажется, будто целая вечность... Так сумел ли он за это время достичь поставленных целей или, быть может, даже вышел на новые рубежи, которые его создатели ранее и представить себе не могли?

А до победы рукой подать...

С позиций дня сегодняшнего ясно видно, насколько хорошо Java в относительно короткий срок сумел зарекомендовать себя и закрепиться на рынке корпоративных систем. Пожалуй, ему нет равных там, где нужно обеспечить долговременную гарантию вложенных инвестиций. Java давно уже стал не просто языком программирования и даже не одной из новых программных технологий — теперь это семейство платформ и целая инфраструктура для создания, развертывания и эксплуатации самых разнообразных информационных систем.

Быстрая адаптация (переносимость, перенацеливаемость) компонентов и самих промышленных систем под новое оборудование и новое операционное окружение, удобство коллективной разработки больших программных комплексов и последующего сопровождения программного кода и, наконец, огромная информационная, интеллектуальная и технологическая поддержка в индустрии, научной и образовательной среде — вот огромные плюсы Java. Именно они и оказываются нередко определяющими при выборе языка реализации для серьезных корпоративных систем.

Казалось бы, язык Java вступил в свою золотую пору и может в полной мере пожинать плоды колоссальных маркетинговых усилий корпорации Sun Microsystems и ее партнеров, прежде всего IBM. Но в действительности дело обстоит не так просто. Невольно в памяти всплывает рекламный ролик, показанный корпорацией Sun на Всемирном форуме JavaOne'98 в Сан-Франциско под названием «Java на марше». Стилизованный под кадры фронтовой кинохроники начала Второй мировой войны, он выглядел устрашающе. Диктор зачитывал победные реляции вермахта с европейских фронтов 1939—1940 гг., перемежая их с отчетами о победах Java на компьютерных фронтах. По замыслу устроителей такая демонстрация должна была, видимо, убеждать в мощи и непобедимости нового сверхсекретного оружия Sun. Но это оставляло какое-то двойственное ощущение, ведь все знают, с чего начинал Третий рейх, и не хуже известно, чем он закончил. Так что, по-моему, приведенная аналогия здесь не срабатывает. Очень бы не хотелось, чтобы Java была уготована подобная участь.

Ключевые вехи развития Java

Рождение Java окутано ореолом романтики. В январе 1991 г. стартовал проект Behind the Green Door, получивший также название The Green Project (http://java.sun.com/people/jag/green). Его целью было создание сверхпортативного компьютера, который играл бы роль универсального дистанционного пульта управления для бытовых устройств [1]. Данную идею не удалось реализовать, и хотя продукт был создан, по экономическим соображениям он на рынок не попал. Но как это нередко бывает, попутно были разработаны такие технологии, ценность которых оказалась значительно выше, чем самого конечного продукта. Для программирования экзотического устройства на базе набора микросхем MicroSPARC с 4 Мбайт оперативной памяти, именуемого *7 (Star Seven), требовалась своя операционная система, и впоследствии ею стала Green OS. Тогда со всей остротой встал вопрос: на каком языке ее реализовывать?

Первоначально Патрик Нотон и Джеймс Гослинг, главные участники команды разработчиков, насчитывавшей 15 человек, планировали создать интерпретируемый диалект крайне популярного в те годы языка Си++. Однако Гослинга, отвечавшего за этот фронт работ, смутила ненадежность и запутанность конструкций языка. Поэтому было решено создавать свой язык, получивший название Oak.

Гослинг писал: «Мы хотели построить систему, которая могла бы быть легко запрограммирована без несметного количества тайных заклинаний и облегчила бы решение сегодняшних задач в стандартной повседневной практике... Когда мы начинали проект, то собирались использовать Си++, но столкнулись с множеством проблем. Сначала это были просто технологические трудности реализации компилятора, но по прошествии времени перед нами встали проблемы, которые лучше всего было решать путем изменения языка» [2].

К концу лета 1991 г. работа по созданию исполняющей системы языка Oak была завершена. Так был получен первый работающий прототип Java (хотя формально рождение языка Oak все же относят к 1992 г.). Здесь пригодился опыт Гослинга в реализации p-кода (виртуальной Паскаль-машины). В 1975 г. Гослинг вместе с Недом Китлицем и Бобом Сайдботемом участвовал в построении среды программирования Pyxis/Multics Pascal, способной по быстродействию кода и удобству интеграции на равных конкурировать в Multics с родным для этой ОС языком PL/I. А начинали они с поддержки компилятора ETH/Zurich Pascal, разработанного в Цюрихе группой профессора Никлауса Вирта [3]. В 1979 г. Гослинг реализовал PERQ — транслятор с p-кода (переносимый Паскаль-код, созданный группой Вирта, прообраз байт-кода Java) в машинный код DEC VAX. Вот, оказывается, откуда берет истоки современная архитектура виртуальной Java-машины!

Но вернемся к проекту Green, формально окончившемуся 3 сентября 1992 г. Хотя он в итоге (после превращения в FirstPerson, 1992—1994 гг.) и завершился коммерческой неудачей, Гослинг продолжал упорно совершенствовать свое детище в надежде применить его для какой-нибудь стоящей задачи. В июне 1993 г. Марк Андриссен и Эрик Бина в Национальном суперкомпьютерном центре США разработали первую версию браузера Mosaic. Это стало катализатором освоения Интернета и появления полноценного Web-интерфейса. Во многих университетах и исследовательских центрах мира закипела работа. Так, в Швейцарском федеральном технологическом институте (ETH) под руководством Никлауса Вирта и Юрга Гуткнехта в то время близился к завершению стартовавший в 1985 г. проект Oberon (http://www.oberon.ethz.ch). Он стал источником многих плодотворных идей, две из которых привели к зарождению Java. Это встроенный Web-браузер системы Oberon с поддержкой аплетов и динамическая компиляция (code-generation on-the-fly) Михаэля Франца (http://www.ics.uci.edu/~franz), положенная в основу нынешних JIT-, AOC— и DAC-компиляторов Java [4].

В марте 1994 г., сразу после защиты в ETH своей диссертации, Михаэль Франц выступал с докладами по системе Oberon и динамической компиляции в Sun Laboratories. Практически сразу после этого Билл Джой, вице-президент Sun Microsystems, став одним из первых обладателем лицензии на ETH Oberon, принял судьбоносное решение о переориентации Oak на Интернет [5]. А осенью того же года в корпорации Sun был разработан браузер WebRunner. Затем Артур ван Хофф переписал компилятор Oak (написанный Гослингом на Си) на самом Oak. И уже в начале 1995 г. язык Oak был переименован в Java, а браузер WebRunner — в HotJava. Так родилась технология Java...

По сути, историю Java можно разбить на три этапа:

- зарождение языка и технологии (1991-1995);
- формирование инструментария и базовой архитектуры (1996-1998);
- ориентация на корпоративный рынок и сегментация Java-платформ (1998-2001).

Сейчас начинается четвертый этап: создание инфраструктуры Web-сервисов с активным развитием встроенных систем (табл. 1).

Таблица 1.

Дата	Рынок	Событие
1991, 15 января	M	Начало проекта The Green Project
1992	M	Создание языка Oak (Джеймс Гослинг)
1994, май	S	Oak переименовывается в Java и позиционируется как язык для Интернета (идея Билла Джоя) с использованием аплетов
1995, 23 мая	S	Джон Гейдж (Sun) и Марк Андриссен (Netscape) официально представляют Java на выставке SunWorld Expo'95
1996, январь	S	Выпуск JDK 1.0 (Java Development Kit)
1996, январь	E	Анонс технологии сервлетов Java ("аплеты для Web-серверов")
1996, май	M	Анонс JavaOS (операционной системы для бытовой электроники)
1996, октябрь	S	Анонс первого JIT-компилятора: HotSpot (Sun)
1996, октябрь	M	Анонс спецификации JavaCard (для программирования смарт-карт)
1997, январь	S	Выпуск JavaBeans JDK (инструментарий для компонентного ПО)
1998, март	E	Анонс Enterprise JavaBeans (JavaBeans для корпоративных систем)
1998, декабрь	S	Выпуск Java 2 Platform (второе поколение языковой платформы Java)
1999, январь	M	Анонс технологии Jini (сетевая интеграция разнородных устройств)
1999, июнь	E	Представление JSP (Java Server Pages, генерирование Web-страниц)
1999, июнь	E, S, M	Анонс платформ J2EE (Enterprise Edition), J2SE (Standard Edition) и J2ME (Micro Edition)
1999, декабрь	E	Выпуск платформы J2EE
2000, февраль	E	Выпуск Java API for XML (программный интерфейс к XML)
2001, февраль	E	Анонс стратегической инициативы Sun ONE (Open Net Environment)
2002, январь	M	Выпуск спецификации Real-time Specification for Java 1.0 (RTSJ)
2002, июнь	E	Выпуск платформы для построения Web-сервисов (Sun ONE Developer Platform)

Примечание. E — корпоративный рынок (Enterprise), S — рынок настольных систем (Standard), M — рынок систем реального времени, бытовая электроника (Micro).

Дабы не заниматься таким неблагодарным делом, как попытка пересказать в двух словах историю развития Java, хотелось бы обратить ваше внимание на выстроенные в хронологическом порядке высказывания самого Джеймса Гослинга, автора и идеолога Java, ныне одного из вице-президентов Sun Microsystems (см. «Проблемы и пути развития Java»). Разумеется, подборка составлена субъективно, но, на мой взгляд, четко отражает болевые точки развития новой технологии.

Последнее высказывание Джеймса Гослинга на JavaOne'2002 весьма контрастирует с официальной позицией Sun, заключающейся в направлении усилий на корпоративный рынок и дальнейшем их развитии в сторону Web-сервисов. Небольшое отступление. Во время московской конференции Java Technology Conference, прошедшей в середине апреля 2002 г., Стюарт Стерн, вице-президент Sun, был несколько озадачен моей просьбой прокомментировать эти слова Гослинга, поставив под сомнение сам факт подобного интервью и подтвердив официальную позицию корпорации. Так что либо Sun пытается объять необъятное и продолжает вести борьбу на всех фронтах сразу, либо точку зрения Гослинга внутри руководства Sun разделяют далеко не все.

Проблемы Java

Нынешнее положение Java никак нельзя назвать безоблачным — тучи над языком и платформой Java сгущаются с каждым днем. У нее появился очень сильный конкурент. С того момента, как в июле 1999 г. Microsoft объявила о начале работ над COOL, впоследствии положенных в основу анонсированной в июне 2000 г. платформы .NET, прошло всего три года, а чем дальше, тем отчетливее проявляются проблемы Java.

Первое, что бросается в глаза: Java до сих пор не имеет международного стандарта (и это удивительно). И в рамках ISO, и в рамках ECMA соответствующие работы фактически заморожены

(отозваны самой Sun). Имитация же деятельности подобных структур силами конкретной компании (Sun) не может служить полноценной заменой общепринятым нормам регулирования производства. А вот Microsoft, едва анонсировав свои новые технологии, уже успела получить европейский стандарт ECMA и на язык C#, и на языковую платформу CLI (Common Language Infrastructure), лежащую в основе .NET. Можно, конечно, считать это происками конкурентов Sun, оказывающих давление на соответствующие институты стандартизации, но факт остается фактом.

Вторая проблема — переносимость. Ключевой для Java принцип WORA (Write Once Run Anywhere — «Напиши однажды — запускай повсюду») на поверку оказался красивым мифом, что, по сути, и было закреплено самой корпорацией Sun выпуском платформ J2EE (для серверов), J2SE (для настольных компьютеров) и J2ME (для бытовой электроники). Попробуйте, к примеру, ответить на вопрос: можно ли приложение, созданное для персонального компьютера (в рамках J2SE), гарантированно исполнить на встроенных системах (платформа J2ME), если это позволяют аппаратные ресурсы? А как быть с сервлетами (servlet), аплетами (applet) и мидлетами (midlet)? Вспомните про несовместимость разных библиотек (в частности, Sun Swing и IBM SWT), а также самих релизов JDK? Ну, скажете вы, так это понятно — разное окружение, разные библиотеки классов. Java ведь растет, развивается. Но на практике весьма проблематично писать приложения, не пользуясь подобными библиотеками и обходясь только своими. А знаете ли вы, что даже специализированные аппаратные Java-процессоры обладают подобным изъяном совместимости (процессор DeCaf компании Aurora VLSI поддерживает 95% всех возможных инструкций байт-кода Java, а процессор Jazelle компании ARM Ltd. и того меньше — 68%)? Если обратить внимание на тот факт, что под каждой из трех базовых Java-платформ лежат разные ОС, то становится очевидно, что Java не позволяет полноценно решить проблему переносимости ПО между серверами, настольными компьютерами и бытовой электроникой. Что касается переносимости внутри каждой из этих групп, то для серверов и настольных компьютеров, где доминируют всего несколько семейств ОС, ситуация почти благополучная. Про встроенные системы, системы реального времени и бытовую электронику, где разнообразие родов и видов напоминает иные заповедники, такого сказать нельзя. А ведь это миллионы и миллионы смарт-карт, мобильных телефонов, карманных компьютеров, различных датчиков и других миниатюрных устройств.

Третья проблема — недетерминированность исполнения программного кода. Если обычная программа выполняется в машинных инструкциях, и тогда на ее различие в параметрах времени и в поведении (при использовании процессов) оказывают влияние только аппаратура и ОС, то для Java добавляется еще виртуальная машина (JVM) со встроенным JIT-компилятором. Все производители по-разному реализуют динамическую генерацию кода (частично код выполняется самим интерпретатором JVM), к тому же одно из главных достоинств Java — автоматическая сборка мусора — требует соответствующей платы за удобство: подчас значительные издержки производительности да еще в непредсказуемое время.

Следующая проблема — обеспечение требуемой производительности. Уж вроде бы столько усилий было потрачено на совершенствование техники динамической компиляции Java-программ, а трудности как были, так и остаются по сей день. Почему же производительность Java на корпоративном рынке вполне приемлема, а вот в области настольных систем и бытовой электроники ситуация по-прежнему довольно непростая? Частично ответ на этот вопрос дает один из лидеров рынка серверов приложений — компания BEA. По данным ее исследований [6], если для клиентской JVM на долю трансляции и выполнения байт-кода приходится 75% времени (еще 20% — на сборку мусора и 5% — на обработку ввода-вывода и процессов — threads), то для серверной JVM картина выглядит иначе: 60% примерно в равной пропорции приходится на процессы и ввод-вывод (сокеты, файлы, БД), 25% — на генерацию кода и 15% — на сборку мусора. Следовательно, небольшая доля времени генерации кода в случае серверной JVM снимает присущие виртуальным машинам проблемы снижения производительности. Да и сама реализации JVM для применения на серверах должна отличаться от реализации для настольных систем (это подтверждают и исследования IBM Tokyo Research Lab). Что касается быстродействия в рамках J2ME, то весной этого года были обнародованы первые результаты создания корпорацией Sun виртуальной Java-машины в рамках проекта Monty (с прицелом на использование для Palm и Pocket PC). По словам Кеннета Толмана (Sun), быстродействие стало в 8—10 раз выше, чем у K-машины Java (K Virtual Machine) версии 1.03, ныне применяемой в J2ME. Ну а если познакомиться с работами компании Esmertec (она отпочковалась от Oberon microsystems, ранее созданной при участии Н. Вирта из сотрудников ETH) и ее инструментарием Jbed, то легко понять, насколько лидеры индустрии «запустили» ситуацию с Java на этом рынке.

Наконец, еще одна важная проблема — моноязыковая среда. Каким бы замечательным ни был язык Java, не стоило искусственно отгораживать его от огромного разнообразия современных языков программирования. Это было одной из явных ошибок Sun (по мнению Гослинга, высказанному в январе 2001 г., на базе JVM разумно развивать целое множество языков). А ведь JVML (язык Java-машины, т.е. байт-код Java) по своей сути уже задавал определенную основу для интеграции разных языков. Это нашло отражение в различных компиляторах, созданных в основном руками энтузиастов в

университетах и небольших компаниях (см. «Мир ПК», № 8/02, с. 124—125). Корпорация Microsoft, идя по стопам своего конкурента, учла просчеты Sun и спроектировала .NET как мультиязыковую платформу. Визави JVML, язык MSIL (Microsoft Intermediate Language), в сочетании с CTS (Common Type System), VOS (Virtual Object System) и другими составляющими Microsoft CLI стал значительно более эффективным инструментом. Он несущественно отличается по своей структуре от JVML, но поддерживает разные языки программирования. Теперь языки объединяются не просто универсальным промежуточным кодом, а единой системой типов, единой системой наследования и унифицированным механизмом обработки исключений, пронизывающим всю платформу сверху донизу.

Мобильность кода и динамическая компиляция

Как показало время, язык Java не стал калифом на час, а занял достойное место на Олимпе языков программирования. Гослинг создавал свое детище на базе кардинальной ревизии Си++, объектной идеологии своего любимого языка Simula-67, виртуальной Паскаль-машины, средств работы с классами языка Objective C (разработанного в NeXT), модульности Mesa (Xerox PARC) и Modula-2 (ETH), механизмов сборки мусора, поддержки процессов и обработки исключений языка Modula-3 (DEC SRC). Все эти очень интересные языки, кроме Си++, практически не известны широкой аудитории. Неудивительно, что созданный на их идеях Java нашел огромную поддержку в мире разработчиков. По оценке Gartner, число Java-программистов достигло 2,5 млн, а по оценке Sun, их больше 3 млн. Компания IDC предсказывает, что к 2003 г. число Java-программистов возрастет до 4 млн.

Михаэль Франц пишет: «Как ни смешно, маркетинговая машина Sun сделала для нас, преподавателей информатики, то, чего мы безуспешно добивались не один десяток лет: даже законченные хакеры приобщились теперь к добродетелям типизации и абстрагирования данных» [5]. Сейчас уже около 78% университетов, готовящих ИТ-специалистов, обучают языку Java.

Выскажу спорное утверждение: главное потенциальное достоинство Java — это не собранные в одном месте новации языков программирования и не обширная инфраструктура для построения сложных систем. Основным является мобильность кода, т.е. возможность пересылать по сети программные фрагменты, которые на лету собираются и выполняются на значительном числе локальных компьютерных платформ (от суперкомпьютеров до кофеварок). В эпоху мобильных систем и развития беспроводной связи это становится особенно важным.

К сожалению, в области Web-интерфейса технология Java в мобильном исполнении в виде аплетов не оправдала искусственно завышенные ожидания. Неудивительно, что теперь эту нишу пытаются занять другие, в частности технология Curl (http://www.curl.com).

Мобильность кода подразумевает решение двух задач: его доставки и исполнения. Первая в идеале связана с организацией специальной динамически реконфигурируемой среды распределенных вычислений, вторая — с исполнением программного кода на «чужой» платформе.

Перенастраивать на лету машинный код одной платформы под другую — дело не только весьма сложное, но и крайне затратное с точки зрения машинных ресурсов. Поэтому остается лишь держать код в некоем промежуточном представлении, наиболее оптимальном как по компактности, так и по скорости его трансляции. С 1960-х годов чаще всего идут по пути наименьшего сопротивления — реализовав виртуальную машину, интерпретирующую промежуточный код и служащую исполняющей прослойкой между программой и операционной платформой.

Кардинально иной и притом крайне простой подход к достижению переносимости предложил и реализовал в 1992—1994 гг. Михаэль Франц (ETH) в рамках технологии OMI для системы Oberon. Компилятор обычно делится на две части: front-end и back-end. Первая отвечает за сканирование исходного текста и формирование промежуточного представления, вторая занимается оптимизацией и генерированием объектного кода под целевую аппаратную платформу. Франц предложил заканчивать компиляцию на первой фазе, а результат сохранять в виде особым образом сжатых файлов (древовидная структура на основе семантического словаря); их можно размещать на диске и передавать по сети. Вторую часть компилятора он соединил с загрузчиком. В результате кодогенерирущий загрузчик так быстро формировал машинный код (для Macintosh), что потери времени по сравнению с обычным запуском составляли около 10—20%. Если же принять во внимание, что быстродействие процессоров с каждым годом растет более высокими темпами, нежели время доступа к устройствам хранения, то станет очевидна вся перспективность подобного решения.

Вот что пишет Франц в своей диссертации: «... такая система открывает несколько абсолютно новых возможностей, в числе которых перспектива создания целой индустрии, производящей компоненты программного обеспечения для пользователей, причем эти компоненты можно применять

непосредственно на нескольких типах целевой архитектуры». Из работы Франца следует еще один важный вывод: на Oberon (читай — Java) свет клином не сошелся. Практически того же самого можно добиться и для ряда других языков подобного типа.

Как известно, если имеется M языков и N целевых аппаратных платформ, то вместо создания MiN полноценных компиляторов достаточно иметь всего M компиляторов промежуточного кода (front-end) и N генераторов машинного кода (back-end). Идея такого «языкового коммутатора» (linguistic switchbox) была материализована еще в 1958 г. Для этого промежуточного языка было выбрано имя UNCOL, что означает «универсальный машинно-ориентированный язык» (universal computer-oriented language). Дух языка UNCOL по-прежнему живет во многих семействах компиляторов. Разумеется, подход Франца возник не на пустом месте. Он соединил идею UNCOL с ANDF-форматом представления промежуточного кода и введением этапа кодогенерирующей загрузки. ANDF-формат (Architecture Neutral Distribution Format) создан военным ведомством Defence Research Agency в Великобритании и опирается на промежуточный язык TDF с древовидной структурой кода.

С точки зрения классификации Java-компиляторов (табл. 2) [7], подход Франца относится скорее к AOT-компиляции. Этим приемом ныне пользуется Jbed Fast Byte Code Compiler швейцарской компании Esmertec для платформы J2ME. Время компиляции у него сопоставимо со временем верификации байт-кода, выполняемой при запуске интерпретаторов виртуальной Java-машины.

И все же Гослинг и Sun пошли по иному пути, потянув за собой всю индустрию. Из всех типов компиляторов для Java наибольшее распространение получили JIT-компиляторы. Они встраиваются в виртуальную машину и включаются в работу (генерируют и записывают в память машинный код, а затем просто передают на него управление) только после того, как зафиксировано неоднократное исполнение одного и того же участка программного кода. Обычно гранулирование участков ведется на уровне метода. Сколько раз до этого код должен быть исполнен интерпретатором, зависит от конкретного JIT-компилятора. Вот где ахиллесова пята JIT-подхода. Производительность падает и за счет постоянного переключения между интерпретатором и JIT-компилятором. Еще один важный недостаток JIT-подхода — высокие требования к объему оперативной памяти.

Менее распространена особая форма JIT-компиляции, получившая название адаптивной компиляции (DAC). Во время выполнения программного кода накапливается важная информация, обеспечивающая наивысшую оптимизацию кода, сборки мусора и управления процессами. Проблема лишь в том, что пока все это накопится и обработается, поезд уйдет далеко и надобность в исполнении таких блоков отпадет сама собой. Вот почему DAC более подходит для предварительного прогона программы с последующим запуском сохраненного машинного кода.

Пожалуй, одними из самых эффективных в отрасли JIT-компиляторов Java (не считая Sun HotSpot) являются JRockit компании BEA Systems и IBM JIT Compiler. Что касается открытых проектов, то тут вне конкуренции Jikes Research VM (IBM), написанный на Java и ставший преемником компилятора Jalapeno [8].

Таблица 2. Типы компиляции, используемые для Java

Тип	Название	Компиляция	Характер	Генерация кода	Наличие Java VM
WAT	Way Ahead of Time Compilation	Полностью	Статическая	До загрузки	Нет
AOT	Ahead of Time Compilation	Полностью	Динамическая	В момент загрузки	Нет
JIT	Just-in-Time Compilation	Частично	Динамическая частичная	Во время выполнения	Да
DAC	Dynamic Adaptive Compilation	Частично	Динамическая адаптивная	Во время выполнения	Да

Будущее Java

Появление весной 2002 г. инструментария Visual Studio .NET, предназначенного для разработки приложений под платформу Microsoft .NET, стало мощным допингом для компаний, занимавшихся развитием технологии Java. Дело не ограничивается традиционной вотчиной Microsoft — семейством Windows (включая Windows CE). Полным ходом идут работы по переносу .NET силами сторонних компаний на Linux и FreeBSD. Очевидно, что с технологической точки зрения .NET превосходит Java 2 Platform. Такой вывод можно сделать, даже не вдаваясь в технические детали, а просто обратив внимание на три момента: заметное оживление и «толкание локтями» в стане лидеров Java-

индустрии, проектирование .NET на основе анализа недостатков Java, а также привлечение к работам на стадии проектирования платформы таких авторитетных специалистов, как Бертран Мейер (автор языка Eiffel, ныне перешедший в ETH), Юрг Гуткнехт (автор Oberon, ETH), Клеменс Шиперски (идеолог компонентной архитектуры в Oberon, перешел из ETH в Microsoft Research), Андерс Хейльсберг (автор Turbo Pascal и Delphi, перешедший в Microsoft из Borland). Microsoft поддержали как минимум две крупные компании: Rational и Borland выпустили инструментарий (Rational XDE и Borland Delphi 7 Studio) для поддержки как J2, так и .NET. Хотя справедливости ради надо отметить, что полнота поддержки .NET сторонними по отношению к Microsoft компаниями чаще всего лишь декларируется. Обычно дело ограничивается возможностью реализации Web-служб. Иногда включаются в инструментарий и компиляторы с генерацией MSIL-кода (например, в Delphi 7 Studio).

В любом случае пока .NET еще только вышла на старт. Потребуется не один год, чтобы сформировалась армия разработчиков, распространилась по миллионам компьютеров весьма объемная среда поддержки .NET (сейчас это около 30 Мбайт), а также созрела сама платформа.

Пользуясь тем, что Sun, почивая на лаврах, позабыла истоки Java и «забросила» рынок бытовой электроники, Microsoft предпринимает в этом направлении все более агрессивные шаги. Как показывает анализ эволюции Java, для корпорации Sun маркетинговая борьба и прямое столкновение с Microsoft стоят на первом плане, в то время как сырые технические решения Sun доводят до ума, как правило, другие компании (прежде всего IBM и Borland Software).

Сейчас Sun и IBM столкнулись на почве мультиязыкового Java-инструментария, противопоставляемого Visual Studio .NET. Корпорация Sun продвигает NetBeans (http://www.netbeans.org), лежащий в основе его линейки продуктов Forte. Тогда как IBM, понимая нешуточный характер борьбы за миллионы разработчиков, серьезно вкладывается в систему Eclipse (http://www.eclipse.org). Причем Eclipse не только отличается открытостью и подбором языков (Java, Си++, HTML, JSP, в планах — Perl, Python, Ruby, Lisp, Scheme, XML, и др.), но и тем, что исповедует новую модель продвижения продукта с ограничением использования, которая в противовес «open source» носит название «open choice» («открытый выбор»). Инструментарий Eclipse берет свое начало в Envy — среде программирования языка Smalltalk, которая четыре года назад была превращена в Visual Age for Java. Теперь все продукты IBM Visual Age делаются на базе Eclipse. Для его развития образован консорциум eclipse.org, куда вошли IBM, Borland, Hitachi, Fujitsu, Rational Software, RedHat, SuSE, QNX Software Systems.

Интересно, что в битве за Java монстры активно скупают небольшие высокотехнологичные компании. Так, корпорация Sun заполучила технологию NetBeans от чешской компании NetBeans Ceska Republika a.s. Корпорация IBM стала владелицей Eclipse, купив канадскую компанию Object Technology International. BEA Systems сейчас обладает очень мощной серверной JVM (JRockit), поскольку приобрела шведскую компанию Appeal.

Каким видится будущее Java? Где же развернется генеральное сражение между Sun и Microsoft? Какова в нем будет роль IBM, Borland, Rational и других лидеров ИТ-индустрии?

В одном из своих интервью, данных во время последнего Всемирного форума JavaOne'2002, Джеймс Гослинг, отвечая на вопрос о том, в каком направлении будет развиваться технология Java в ближайшие пять лет, произнес следующее: «Я думаю, наиболее интересное направление — это по-настоящему повсеместное использование технологии Java во всех видах ключевых мест, которые расположены на концах сети. В области корпоративного рынка уже реально создано множество инфраструктурных средств, но существует и немало вновь создаваемых продуктов, где используется технология Java. Сотовые телефоны, автомобили, системы реального времени — все это крайне интересно и дает людям возможность строить системы путем сборки отдельных элементов, что, как мне видится, очень притягательно».

Это не просто слова. Два последних года жизни отдал Гослинг разработке спецификации Java реального времени (Real-time Specification for Java). Интересно, что она создавалась при активной поддержке со стороны IBM.

Гослинг, вне всяких сомнений, видит реальную угрозу со стороны Microsoft на тех рынках, которые Sun ранее считала для себя неинтересными (а с какой идеи началось продвижение Java, похоже, забыли и в самой Sun). Теперь же начинает набирать силу новое модное веяние (X Internet), способное приносить хорошие деньги. Соглашения Sun и Borland с ведущими производителями на рынке мобильной связи в отношении внедрения J2ME в сотовых телефонах дают определенную фору Sun перед Microsoft. Borland Software, особенно с выходом ее новой среды разработки JBuilder 7, становится на этом рынке одним из явных лидеров.

Стоит упомянуть, что существуют и направления совместного использования Java для мобильных телефонов и серверов. Сейчас, в частности, ведутся работы по формированию архитектуры OMA

(Open Mobile Architecture) для систем подобного класса. Среди инициаторов OMA такие лидеры индустрии, как IBM, Sun, BEA, AT&T, Nokia и NTT [9].

Ключевым в битве гигантов наверняка станет развитие Web-сервисов (прежде всего коммерческих, требующих особого внимания к безопасности) в контексте Java 2 Platform и .NET. Кто первым успеет застолбить новую инфраструктуру, средства ее поддержки и разработки, тот и получит мощнейший козырь в борьбе за передел практически всех рынков. Судьба Java во многом будет зависеть от исхода этого сражения.

Литература

1. Шершульский В. Неизвестные страницы истории языка Java // ComputerWeek Moscow. 1996. № 18.
2. Gosling J. Java: An Overview // Sun Microsystems, February, 1995.
3. Богатырев Р. Летопись языков. Паскаль // Мир ПК. 2001. № 4.
4. Богатырев Р. Гадание на кофейной гуще // Мир ПК. 1998. № 2.
5. Франц М. Java: критическая оценка // Мир ПК. 1997. № 8.
6. BEA WebLogic JRockit — The Server JVM. Increasing Server-side Performance and Manageability // BEA Systems, 2002.
7. Dickman L. A Comparison of Interpreted Java, WAT, AOT, JIT, and DAC // Esmertec, 2002.
8. The Jalapeno Virtual Machine // IBM Systems Journal. 2000. Vol.39. № 1.
9. Lawton G. Moving Java into Mobile Phones // IEEE Computer, June, 2002.

Проблемы и пути развития Java

Из интервью Джилл Стейнберг и Билла Веннерса (JavaWorld) с Джеймсом Гослингом (Sun Microsystems) на Всемирных форумах JavaOne (1998—2002)

JavaOne'98 (март)

Корпоративные системы. Ответ на вопрос о том, чем в основном отличаются акценты нынешнего форума JavaOne от сделанных на прошлогоднем: «Самое важное то, что Java сейчас уже не просто игрушка, с которой возятся родители. Если вы посмотрите на банки и крупные финансовые учреждения, то обнаружите, что они решают большие и важные задачи, критические для бизнеса. Сегодня они уже не забавляются, а делают реальные деньги и имеют реальный успех».

JavaOne'99 (май)

Производительность. Ответ на вопрос о проблемах производительности Java в связи с выходом Java HotSpot: «Производительность — это одно из того, чего никогда нельзя завершить. На Земле нет ни одной системы, для которой достигнута наивысшая производительность. Надо сказать, что производительность Java довольно неплохая... Мы находимся в этом плане на одном бейсбольном поле вместе с Си и Си++».

JavaOne'2000 (май)

Детерминированность. Ответ на вопрос об особенностях работы Java в системах реального времени: «В мире таких систем наиболее критичным аспектом является производительность. Но еще важнее — детерминированность.

Когда вы говорите: «Я хочу, чтобы то-то было выполнено в определенное время», то обычно и думаете об этом, а затем следите за этим же. Здесь можно выделить две стороны. Одна — диспетчеризация... Разрабатываемая спецификация для Java реального времени (realtime Java) задает основу для формирования диспетчеров самого экзотического характера. Другая сторона — сборка мусора, там, где для Java присутствует наибольший недетерминизм в смысле времени. Это самая тяжелая проблема».

JavaOne'2001 (июнь)

Сложность. Ответ на вопрос, не изменил ли Гослинг свой взгляд на то, что сложность — главный вызов программистам: «В определенном смысле это то, над чем я сейчас работаю. Как писать сложное приложение? Как иметь дело с приложением в миллион строк исходного текста? Как найти ключ к тому, чтобы его понять? Как вносить изменения в системы подобного масштаба? Как совладать с ними? Другая ось сложности проходит через распределение приложения по сети. Один из тех вопросов, где Java силен, — способность дать однородный взгляд на ту реальность, которая на самом деле неоднородна».

JavaOne'2002 (март)

Системы реального времени. Ответ на вопрос о повсеместном увлечении разработкой серверного ПО и игнорировании Java в эпоху бурного развития смарт-карт и мобильных телефонов: «Я думаю, это взгляд со стороны американского рынка. Если вы приедете на конференцию в Северную Америку, то там люди будут говорить о корпоративном ПО. Я недавно побывал на Java-встречах в Европе и Японии – и никто не говорил о корпоративных системах. Что же всех волнует? Устройства и мобильные телефоны, а также то, каким образом построить единое целое из отдельных компонентов. Американским журналистам имело смысл попасть на недавно состоявшуюся японскую конференцию JavaOne, где трудно было найти что-либо из корпоративного ПО. Там были представлены встроенные системы и системы реального времени – одни неудачные, другие – превосходные. Это и есть воплощение идеи повсеместных вычислений».

Из ответов Джеймса Гослинга на вопросы аудитории по линии Java Developer Connection Program (март 2002 г.)

Вопрос: «Язык Java с каждым новым релизом добавляет новые средства. В принципе это хорошо, но в результате он становится достаточно большим. Если бы у вас была возможность изъять некоторые вещи, то что бы вы сделали?»

Ответ: «Сам язык Java фактически добавляет не очень много средств. Язык Java весьма и весьма стабилен, и лишь небольшое число новых средств включается в каждый новый релиз... Меня спрашивают: если бы вы могли удалить некоторые вещи из платформы J2SE, то что бы это было? Одна из трагедий, с которой мы сталкиваемся, состоит в том, что у нас много потребителей и все, что заключено в платформе, критично для достаточно большого числа людей. Поэтому каждому конкретному человеку, каждому отдельно взятому программисту необходимо далеко не все из платформы J2SE. Причем разные разработчики интересуются различными частями платформы. У нас в Sun ведется постоянная борьба с внесением новых средств. Я, в частности, никогда не работаю с JDBC (Java Database Connectivity) и никогда не пишу кода, использующего средства работы с базами данных. Поэтому если бы вы удалили JDBC из платформы J2SE, то это никак не затронуло бы написанный мною код. Но при этом немалое число других людей были бы огорчены».

Java на рынке бытовой электроники. Факты и цифры

- В 2001 г. было реализовано 14,1 млн. Java-устройств.
- Свыше 200 компаний занимаются поставками Java-приложений для беспроводной связи.
- 15 ведущих производителей сотовых телефонов в ближайшее время готовят к выпуску 50 моделей Java-телефонов.
- К концу 2002 г. планируется продать более 100 млн. Java-устройств, из них около 50% приходится на долю компании Nokia.
- В 2003 г. Nokia планирует довести выпуск своих сотовых телефонов с поддержкой Java до 100 млн.
- К 2004 г., по оценке Sun, количество Java-устройств в мире должно подойти к рубежу 700 млн.
- По прогнозу Gartner, к 2006 г. как минимум 80% мобильных телефонов будут поддерживать Java. Язык Java станет стандартом де-факто на рынке мобильных телефонов среднего и высшего класса.