Руслан Богатырев

1988 — 2003: эволюция программирования за последние 15 лет

Источник: Мир ПК, #10/2003

Полтора десятилетия: много это или мало для программирования? Если вести отсчет от 1952 г., когда в швейцарском Базеле была издана работа Х. Рутисхаузера «Автоматическая разработка плана с помощью программно-управляемых вычислительных машин», то они составляют почти 1/3 той временной шкалы (табл. 1), где рука об руку шли языки программирования (табл. 2) и операционные системы. В этот период вместилась целая эпоха персональных компьютеров, Интернета и КПК, эпоха графического интерфейса и сценарных языков, эпоха доминирования объектно-ориентированного программирования (ООП).

Таблица 1. Три эпохи программирования.

Годы	Языки программирования	Операционные системы
1952 — 1966	Фортран, Алгол-60, Кобол, Лисп, Бейсик, APL, PL/I	IBM OS/360, IBM OS/370
1967 — 1987	Simula-67, Алгол-68, Форт, Пролог, Паскаль, Си, Ada, Modula-2, Smalltalk, C++, Eiffel	DEC RT-11, DEC RSX-11, DEC VAX/VMS, UNIX, CP/M, MS-DOS
1988 — 2003	Оберон, Visual Basic, Java, C#, Perl, PHP, Python	Windows, OS/2, Mac OS, Linux, Palm OS, Pocket PC

Чего же удалось добиться за эти годы, насколько мы приблизились к заветной цели и что нас ждет в будущем? Думаю, ответы на эти вопросы можно найти в высказываниях тех, благодаря кому компьютерные науки и программная инженерия достигли наивысшего расцвета в своей истории. Вот как эти выдающиеся деятели оценивают прошлое, настоящее и будущее программирования.

Таблица 2. Хроника развития языков программирования (1988—2003).

Год	Язык	Год	Язык	Год	Язык
1988	Оберон	1995	Delphi	1999	Haskell 98
1988	Modula-3	1995	Eiffel 3	2000	C#
1988	Borland Object Pascal	1995	Self 4.0	2000	Java 2 (1.3)
1988	Perl 2.00	1995	Sather 1.1	2000	Python 2.0
1988	Tcl/Tk	1995	PHP/FI	2000	PHP 4.0
1989	ANSI C (C89)	1995	LiveScript	2001	C# (ECMA)
1989	Eiffel 2	1995	JavaScript	2001	Delphi 6
1989	Perl 3.00	1996	Eiffel 4	2001	Cobol 2002 (draft)
1990	ISO C (C90)	1996	APL 96	2002	Visual Basic .NET
1990	Haskell 1.1	1996	PostScript Level 3	2002	Delphi 7
1990	Scheme IEEE	1996	JScript	2002	Java 2 (1.4.1)
1991	Oak	1997	Modula-2 ISO	2002	Fortran 2000 (draft)
1991	Oberon-2	1997	Fortran 95 ISO	2002	Perl 5.8.0

1991	Fortran-90 ISO	1997	OO Cobol	2003	C# (ISO)
1991	Visual Basic 1.0	1997	Prolog IV	2003	Zonnon
1991	Python	1997	ECMAScript	2003	Java 2 (1.4.2_01)
1991	Perl 4.00	1997	PHP 2.0	2003	Python 2.3
1992	PostScript Level 2	1998	C++ ANSI/ISO	2003	PHP 4.3.3
1993	Ruby	1998	Java 2	2003	Ruby 1.8
1994	Common Lisp ANSI	1998	Visual Basic 6.0		
1994	Perl 5.00	1998	PHP 3.0		
1995	Java	1999	ISO C (C99)		
1995	ISO C (C95)	1999	Delphi 5		

Прошлое

Бертран Мейер (автор языка Eiffel и концепции проектирования по контракту): «Анализируя историю развития ИТ за последние несколько десятилетий, нельзя игнорировать то обстоятельство, что если большая часть новаторских решений в 60-е и 70-е годы были предложены вузами, то в 80-х и 90-х годах вклад небольших фирм и исследовательских лабораторий крупных корпораций оказался значительно весомее. Столь чрезмерное обобщение, которое лишь подтверждается редкими исключениями, может обидеть некоторых читателей. Однако было бы сложно найти среди последних достижений неопровержимые эквиваленты таким широко известным решениям, как Паскаль и Modula-2 Никлауса Вирта, операционная система THE Эдсгера Дейкстры, взаимодействующие последовательные процессы и мониторы Тони Хоара, язык Simula, созданный в университете Осло. Эти и другие фундаментальные изобретения периода становления компьютерной отрасли доказали, что университеты могут предлагать не только прекрасные теории».

Никлаус Вирт (автор Паскаля, Modula-2 и Оберон): «В программном обеспечении и в программировании вообще часто употребляется термин «кризис программного обеспечения», о котором впервые открыто заговорили в 1968 г. и который сделал популярным структурное программирование. Был признан тот факт, что сложное программное обеспечение может быть понято только тогда, когда оно упорядочено и структурировано. Разработка огромных систем, где задействованы армии «аналитиков», со всей очевидностью доказывает необходимость координации работ, документирования и соблюдения соглашений, которые должны быть представлены в виде спецификаций интерфейсов. Руководство (management) становится доминирующим фактором, и все упомянутые аспекты так или иначе покрываются новой волной, носящей название «программная инженерия» (software engineering). Все это настоятельно требует по-настоящему профессионального подхода» .

Бьерн Страуструп (автор языка C++): «Я — ярый приверженец идеи объектно-ориентированного программирования и связанных с ним принципов и технологий проектирования, впервые появившихся в языке Simula 67. Однако это не единственный эффективный подход. Многие программистские задачи лучше всего решаются с помощью методов, не укладывающихся в узкое определение понятия «объектно-ориентированный»».

Деннис Ритчи (автор языка Си): «Что меняется, так это роль высокоуровневых языков: по мере роста числа людей, связанных с компьютерами, она становится важнее. Языки, поначалу бывшие лишь симпатичными маленькими инструментами, такие как, скажем, Perl или Python, внезапно переместились ближе к центру мироздания».

Ларри Уолл (автор языка Perl): «Наиболее революционным изменением в проектировании языков сейчас можно считать тот факт, что мы все больше внимания уделяем более крупным, многофункциональным языкам. Это прямой результат широкого распространения компьютеров. Люди не хотят изучать новый ограниченный язык каждый раз, когда им приходится работать с новым интерфейсом. Сейчас успех будет сопутствовать не маленьким, ограниченным, а большим, многофункциональным языкам, которые могли бы использоваться так, как если бы они были маленькими».

Настоящее

Эдсгер Дейкстра (патриарх программирования, автор знаменитой книги «Дисциплина программирования»): «Приходится признать, что главная задача компьютерной науки — «не запутать все до неузнаваемости» — так и не была достигнута. Увы, большинство наших систем слишком сложны, чтобы не тревожиться об их состоянии, они слишком хаотичны и запутанны, чтобы с ними можно было чувствовать себя уверенно и спокойно».

Никлаус Вирт: «Лет 25 тому назад интерактивный текстовый редактор мог быть спроектирован из расчета всего лишь 8000 байт памяти — современные редакторы текстов программ требуют в 100 с лишним раз больше. Операционная система должна была обслуживать 8000 байт, а компилятор умещаться в 32 Кбайт, в то время как их нынешние потомки требуют для своей работы многих мегабайтов. И что же, это раздутое программное обеспечение стало быстрее и эффективнее? Наоборот. Если бы не аппаратура с ее возросшей в тысячи раз производительностью, современные программные средства было бы просто невозможно использовать... Характерной чертой компьютерной индустрии является тот факт, что поставщик, которому удалось первым выбросить продукт на рынок, как правило, получает ощутимые преимущества над конкурентом, чей аналогичный — и лучший по качеству! — продукт появляется вторым. Тенденция принимать первый появившийся продукт в качестве de facto-стандарта — крайне прискорбный феномен, вызванный к жизни все той же спешкой».

Джеймс Гослинг (автор Java) отвечает 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».

Никлаус Вирт: «В действительности беды программной инженерии происходят вовсе не из-за отсутствия инструментов или хорошего менеджмента, а от недостатка технической компетентности. Хороший проектировщик должен опираться на опыт, на строгое логическое мышление и на педантичную точность. Никакая чудесная магия не может помочь. В свете всего этого особенно грустно, что во многих университетских программах по информатике «программированием в большом» (programming in the large) пренебрегают. Проектирование не заняло надлежащего места в программах по подготовке специалистов. Как результат, программная инженерия превратилась в эльдорадо для хакеров. Программировать без царя в голове стало условием профессионального выживания: чем более хаотичной выглядит программа, тем меньше опасность, что кто-то возьмет на себя труд проинспектировать этот код и развенчать как саму программу, так и ее автора».

Будущее

Гради Буч (идеолог объектно-ориентированного проектирования): «Станем ли мы в будущем свидетелями радикальных усовершенствований в процессе программной инженерии? Эл Ахо сомневается в этом (и я горячо поддерживаю его точку зрения): «Поскольку программное обеспечение включает в себя людей, процесс и технологию, мы можем как-то усовершенствовать последние два, но первое остается неизменным. Поэтому я считаю, что нам вряд ли удастся наращивать производительность и качество программного обеспечения теми темпами, которые закон Мура обещает для процессоров, оптики и беспроводных технологий».

Ларри Уолл: «Такое ощущение, что компьютерная наука ищет некую панацею, которая позволит людям писать корректные программы даже не думая. На самом деле нам необходимо научить людей думать правильно. И именно здесь компьютерная культура оказывается несостоятельной. Мы забыли о важности фонетических методов изучения компьютерных языков».

Никлаус Вирт: «Я бы не сказал, что распространившаяся практика ООП реализовала все свои потенции. Наша конечная цель — расширяющее программирование (extensible programming). Под этим я понимаю возможность конструирования таких иерархий модулей, когда каждый

добавляет новую функциональность в систему. Расширяющее программирование подразумевает, что можно добавлять модуль без необходимости вносить какие-либо изменения в существующие модули — не должно быть необходимости даже их перекомпилировать».

Деннис Цикритзис (автор бестселлеров по операционным системам): «Программирование выполняется одним человеком или, по крайней мере, тесно координированной группой экспертов. Таково ли понятие программирования в Интернете? Множество программистов не являются экспертами. Большинство программ не имеют четко определенной среды, в которой они должны работать, или четко определенных спецификаций, определяющих, что же они должны делать. Мы уже перешли к другому виду программирования, нежели «дейкстровский». Следует признать этот факт и изменить свои понятия и парадигмы».

Чарльз Саймони (ведущий программный архитектор Microsoft, руководитель проектов по созданию Microsoft Word и Excel): «Интенциональное программирование (Intentional Programming, IP) — пожалуй, одно из самых замечательных явлений современного программного инжиниринга. IP — не новый язык, как Java, и не новая методика программирования, как объектная ориентация. IP — это просто операционная система для абстракций, новая категория метаинструментария, которая координирует взаимодействие независимо разрабатываемых абстрактных объектов, называемых понятиями (intention), так что программисты смогут помечтать о постоянно расширяемой и совершенствуемой интегрированной среде разработки программ. IP позволяет воплотить эту мечту в жизнь за счет создания программ из экземпляров понятий, точно так же как системы автоматизированного проектирования создают план сложных механизмов из объектов».

Бьерн Страуструп: «Вряд ли из меня получится хороший предсказатель будущего, поэтому, на мой взгляд, не стоит и заниматься этим. Единственное, что следует отметить: все новое гораздо чаще, чем нам хотелось бы, напоминает хорошо забытое старое. Заметьте, Кобол, Фортран и Си по-прежнему остаются ведущими языками».

Об авторе

Руслан Богатырев — научный редактор журнала «Мир ПК», гл. редактор CD-приложения «Мир ПК—диск» и электронного альманаха «Искусство программирования», bogatyrev@pcworld.ru