Developers Shift to Dynamic Programming Languages

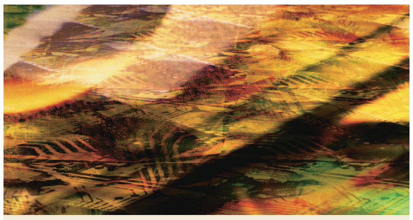
Linda Dailey Paulson

Software developers are always looking for ways to boost their effectiveness and productivity and perform complex jobs more quickly and easily, particularly as projects have become increasingly large and complex.

Programmers want to shed unneeded complexity and outdated methodologies and move to approaches that focus on making programming simpler and faster, said Stephan Deibel, chair of the Python Software Foundation and cofounder of Wingware, a Python development environment vendor.

With this in mind, many developers are increasingly using dynamic languages such as JavaScript, Perl, Python, and Ruby. Although software experts disagree on the exact definition, a dynamic language basically enables programs that can change their code and logical structures at runtime, adding variable types, module names, classes, and functions as they are running. These languages frequently are interpreted and generally check typing at runtime.

Dynamic languages provide an alternative to the more widely used static languages such as C++ and Java. Programs written in static languages have rigid code and logical structures and remain as they are unless the programmer changes them. Static languages generally are compiled, and they check typing in the compilation stage, before the program starts executing.



A survey by Tiobe Software—a software-quality consultancy that maintains an index of programming languages' popularity—says C, C++, and Java are the most popular. However, the survey also indicates that dynamic-language use has increased recently.

This is reflected in the growing use of programming approaches that work with dynamic languages such as AJAX (Asynchronous JavaScript and XML), LAMP (Linux; the Apache Web server; the MySQL database-management system; and Perl, PHP, or Python), and Ruby on Rails.

Also, major companies such as Microsoft and Sun Microsystems are supporting dynamic languages in their development platforms.

Dynamic languages are flexible and can let developers write more to-the-point code quickly and easily, said Python creator Guido van Rossum, who works on development research for Google.

And numerous dynamic languages are open source, which appeals to many developers, noted Carl Howe, principal analyst for Blackfriars Communications, a market research firm, and a longtime Unix kernel coder.

Now, developers are better able to choose dynamic or static languages, depending on which better suits their needs, said Rice University associate professor Dan Wallach.

Despite the enthusiasm, dynamic languages have their drawbacks and concerns, including many developers' unfamiliarity with them.

Moreover, said Kingston University professor Les Hatton, computing has proven to be "a fashion industry with little or no relationship with engineering," and many new programming approaches are "just something new to try before something newer comes along."

INSIDE DYNAMIC LANGUAGES

Typing is the main, but not the only, factor in determining whether a language is dynamic.

Typing

In programming, a type system defines how a language classifies similar types of values and expressions, how it can manipulate them, and how they interact with one another. This generally includes a description of the data structures that the language enables.

Static languages are generally, but not always, strongly and statically typed. Dynamic languages are usually, but not always, weakly and dynamically typed.

Some dynamic languages, such as Lisp, are strongly typed. And some static languages, such as C and C++, are weakly typed.

With *statically typed languages*, programs explicitly declare—but need not define—variables before they are used. Many programmers like the structure that forced variable declaration creates.

With *dynamically typed languages*, programs must define variables—but not explicitly declare them—before they are used. This eliminates the need to write as much code. And many programmers like the ability to flexibly use a variable at will when required, without having to declare it.

Programming languages in which variables are bound to specific data types are considered *strongly typed*. In *weakly typed* languages, variables are not bound to a specific data type, which again provides more flexibility but less structure.

Ins and outs of static and dynamic languages

Strong typing enforces rules as to what programs can do with various elements by detecting and correcting type-related errors at compile time.

Weak typing allows some predefined exceptions to type-related rules or uses a type-violation mechanism, generally for cases in which a specific rule violation won't cause problems for a program. Weak typing corrects the type-related errors for which there are no exceptions at runtime.

It thus avoids the overhead of correcting at compile time the "errors" that won't cause problems for a particular program. However, Wingware's Deibel noted, strong typing detects errors sooner, at compile time, which is more efficient.

Programmer and Ruby on Rails creator David Heinemeier Hansson, a partner in the 37signals Web-application-development firm, said the time it takes for static typing to enforce all rules throughout the process can make it unreasonably slow to get projects to fruition.

Dynamic languages let developers get results quickly, said Blackfriars' Howe.

Their code is more compact because, for example, there is no type-declaration overhead, Deibel explained. This lets developers declare commands efficiently via small amounts of code—rather than detailed, highly specific programming—and quickly produce applications, which is important when an organization needs software to solve an urgent problem.

Python would use 10 to 20 percent of the amount of code that Java or C++ would use to write the same application, according to Deibel.

Being able to write code quickly and efficiently removes some of the detailed and repetitive work from programming and lets developers focus more on creative matters, Hansson said.

However, Hatton contended, many of these languages are insufficiently tested in mission-critical applications for businesses to risk using for these purposes.

And traditionally, dynamic languages haven't performed as well as static languages because of the extra runtime checking. According to Google's van Rossum, "The extra information kept around at runtime to enable the checking by dynamic languages can cause their memory usage to increase."

And the CPU must work harder because it has to evaluate more runtime-checking instructions, according to Jason McConnell, product manager in Microsoft's Developer Division.

However, today's faster processors and larger memory capacities are helping to reduce this concern.

Increased flexibility

Because applications written with dynamic languages don't have to explicitly declare variables and some other structural items before they are used, the programs can introduce variable types, module names, classes, and functions on the fly, thereby providing programmers with flexibility.

However, this also means dynamic languages are not well-suited to rigid types of software development such as system programming, which requires static types and interfaces.

With dynamic languages, developers can easily do things—such as not defining a type for a variable or not defining an interface or abstract class—that are often considered bad practice and that they could not readily do with static languages. With some programs, though, doing these things might be desirable because they would be more efficient and might not cause development-related problems.

Table 1. Tiobe Software listing of most popular programming languages.

Popularity ranking	Language	Static/ Dynamic
1	Java	Static
2	C	Static
3	C++	Static
4	Visual Basic	Static
5	PHP	Dynamic
6	Perl	Dynamic
7	C#	Static
8	Python	Dynamic
9	JavaScript	Dynamic
10	Ruby	Dynamic
11	SAS	Dynamic
12	Delphi	Static
13	PL/SQL	Dynamic
14	D	Static
15	ABAP	Dynamic
16	Lisp/Scheme	Dynamic
17	Ada	Static
18	Cobol	Dynamic
19	Pascal	Static
20	Transact/SQL	Static

DYNAMIC LANGUAGES

Dynamic languages are not new, said Blackfriars' Howe. He noted that languages such as APL and Lisp were developed in the late 1950s. There have been many since then—such as ABAP, Groovy, SAS, and Tcl—with several becoming very popular, as Table 1 shows.

JavaScript and ECMAScript

JavaScript (http://developer.mozilla.org/en/docs/About_JavaScript), based loosely on the strongly typed Java, is a scripting language, also strongly typed, that Netscape Communications created in 1995. It can be embedded into a Web page's HTML code to add dynamic content that can respond to mouse rollovers and other user actions.

ECMAScript, which Europe's ECMA International and the International Organization for Standardization have adopted, is a general-purpose, cross-platform scripting language created to capture the common core elements of JavaScript and Microsoft's comparable JScript.

PHP

In 1995, programmer Rasmus Lerdorf released PHP, the most popular of the dynamic languages, according to Tiobe's survey.

PHP (the initials originally stood for "personal home page") is an open source, server-side, cross-platform, object-oriented scripting language for creating dynamic Web pages. Users insert the PHP code into a page's HTML code.

The PHP Data Objects extension defines lightweight and consistent interfaces for accessing various relational databases.

Microsoft is working with Zend Technologies, a vendor of PHP products and services, to enable the language to run smoothly on Windows.

Perl

Programmer Larry Wall released the Practical Extraction and Reporting Language (www.perl.org) in 1987.

Perl is a Unix-based, open source, interpreted, server-side programming language for writing Web scripts. It was originally designed to process and manipulate text and thus is used extensively with Web-based forms. Developers utilize the language for various tasks, such as Web and GUI development and network programming.

Perl supports procedural, object-oriented, and functional programming approaches, as well as automatic memory management.

Python

Van Rossum created Python (www.python.org) in 1990. It is an open source, interpreted, object-oriented, dynamically typed language similar to Perl. Python—which van Rossum named after the *Monte Python's Flying Circus* TV show—is known for its clear, simple syntax and readability.

The language is highly portable, as a number of operating systems—including the Mac OS, Unix, and Windows—can interpret its statements.

Major companies are supporting dynamic languages in their development platforms

Natively, Python supports structured programming and has features that work with functional and aspect-oriented programming. The language supports other programming approaches via extensions.

Developers, including those for Google, use the flexible Python for creating many types of applications.

Ruby

Computer scientist and programmer Yukihiro Matsumoto publicly released Ruby in 1995.

Ruby—similar to Perl and Smalltalk—is an open source, interpreted, fully object-oriented language known for being clean, concise, consistent, structured, extensible, and portable to multiple operating systems such as Linux, the Mac OS, Unix, and Windows.

To be useful with extreme programming, Ruby lets developers write parts of applications in other languages when that is best for a project.

Ruby is dynamically typed and supports object-oriented, procedural, and functional programming.

INCREASING SUPPORT

Developers have created several increasingly popular development platforms that use dynamic languages. And some large technology companies are including dynamic-language support in their development environments to attract programmers looking for such approaches.

LAMP. Designed to provide a flexible way to develop applications, primarily for the Web, the LAMP platform (www.lampware.org) has become important in the last couple of years because many people are interested in simple Web applications, noted van Rossum. Recently, companies have been using LAMP for more complex programs.

AJAX. Developers are using AJAX—a set of technologies mostly developed in the 1990s—to build Web applications that perform better than traditional Web programs and that look and act more like desktop software.

Seaside. Released in 2004, Seaside (www.seaside.st) is an AJAX-like framework for developing Web applications with the Smalltalk dynamic language.

Ruby on Rails. Ruby is the foundation of Ruby on Rails, a development platform that programmers are increasingly embracing because it is open source and easy to use. Hansson developed Rails, which utilizes integrated programming packages and preset code, known as *conventions*, designed to be complete and ready to work with immediately, without the need for configuration.

Proponents say Rails is best suited to building infrastructures that pull information from a database to a Web application, such as those used in e-commerce, online communities, and data retrieval.

Eclipse Foundation. The Eclipse Foundation—a nonprofit organization for advancing the Eclipse platform of open source, Java-based, software-development tools—is adding a PHP-integrated development environment.

Microsoft. Microsoft is backing

dynamic languages such as IronPython, JScript, and PHP. The company is supporting dynamic languages in areas such as the .NET framework's common language runtime (CLR), which manages the execution of programs written in any of several supported languages.

Meanwhile, Microsoft is hosting the Phalanger compiler project on its CodePlex collaborative-development portal. The compiler lets PHP scripts run on .NET without modifications.

Sun. Of the Java platform's three legs—the virtual machine, the APIs, and the language—the language leg is replaceable, explained Tim Bray, Sun's director of Web technologies. He said Sun will enable Java to support various languages, including dynamic ones.

Currently, the company supports JavaScript in its Java platforms and via its NetBeans integrated development environment. Sun also recently hired the developers of JRuby, a Java implementation of Ruby, according to Bray.

Bray, like many other industry observers and programmers, predicted that dynamic languages will continue developing and growing in popularity and support.

Kingston University's Hatton, however, disagreed, saying that dynamic languages are just the current software-development fashion. "They will appear, hang around for a while, and then disappear," he explained. "This is what happens when fashion dictates progress rather than engineering concepts such as measurement, validation, root-cause analysis, and defect prevention."

Bray noted that many dynamic languages don't have the tools or performance necessary for building large software. But, he added, some of their agility-related features are slowly percolating into enterprise languages such as Java, which will make them more useful.

Howe concluded, "All these languages are a means to an end. They can help some group of people do a job they need to do more easily. Languages are tools for programmers. As the problems change, people use new languages." ■

Linda Dailey Paulson is a technology writer based in Ventura, California. Contact her at ldpaulson@yahoo.com.

Editor: Lee Garber, *Computer*, l.garber@computer.org

ICSC 2007

Sponsored by:

IEEE

IEEE computer society

IEEE First International Conference on Semantic Computing

September 17-19, 2007

Hyatt Regency Irvine, Irvine, California

http://icsc2007.eecs.uci.edu/

The field Semantic Computing applies technologies in natural language processing, data and knowledge engineering, software engineering, computer systems and networks, signal processing and pattern recognition, and any combination of the above to extract, access, transform and synthesize the semantics (contents) of multimedia, texts, services and structured data.

The IEEE International Conference on Semantic Computing (ICSC2007) is an international forum for researchers to exchange information regarding advancements in the state of the art and practice of semantic computing, as well as to identify the emerging research topics and define the future of semantic computing. The technical program of ICSC2007 will consist of invited talks, paper presentations, and panel discussions.

Submissions of high quality papers describing mature results or on-going work are invited. Topics for submission include but are not limited to:

- Natural language understanding and processing
- Understanding and processing of texts and multimedia content
- Content-based retrieval of texts, images, videos and audios
- Speech recognition
- Semantic web search
- Semantic web services
- Semantic annotation of multimedia contents
- Natural language driven computing
- Multimedia driven computing
- Question answering
- Spoken dialogue and multi-modal systems
- Data, knowledge and software engineering issues
- Integration of semantic systems
- Semantic computing and wireless communications
- Content-based security
- Applications of semantic computing
- Hardware support for semantic computing