Alan Knight

Two New Smalltalks

Smalltalk is a very loose term. While there are some clear, defining characteristics, there are also many variations as well. This is important because software development is a large field, with different solutions appropriate to different problems. This article will discuss some of these possible variations, with reference to two promising new implementations.

The first of these new implementations is Object Connect's "Smalltalk MT," a high-performance implementation for Windows NT 4.0, with full compilation and true multithreading. The second is Intuitive Systems' "Dolphin Smalltalk," which boasts a low cost and a very low memory footprint.

Neither one has been officially released yet, but free prerelease copies are available for download from their Web sites (see contact information at the end). Even if you've missed the free periods, both companies are aiming at product prices in the low hundreds (of U.S.) dollars.

Note that I looked at prerelease versions, and that I've only played with them, so all I can do is relay first impressions and generalities about their implementation choices. I hope you'll see a full review in these pages once they are officially released.

COMPILATION

Most current Smalltalks use dynamic compilation, now commonly referred to as Just-In-Time (JIT) compilation. This is an example of a process I call "cross-domain buzzword hybridization." In this process, a recognized buzzword from one domain crosses over to another, where it attaches to a concept almost, but not quite, entirely unlike the original. If you don't like it, think of it as payback for the term "object-oriented."

In dynamic compilation, the methods are stored in bytecode form. At runtime, some (or all) of these bytecodes are translated into machine code. This can provide most of the speed of machine code with a much smaller executable size, comparable to that of bytecodes.

Although this is a good approach, it's not a perfect solution for every circumstance. It takes more space than bytecodes (you now have to store both compiled forms) and still won't run as quickly as compiled code (you have to spend CPU cycles to translate, and you can't afford to think too hard about optimization). Finally, it requires a VM to do the translating, which makes it more difficult to create standalone executables, DLLs, and callback functions.

Smalltalk MT is fully compiled, and has "an optimizing compiler that generates fast, compact code." Full compilation should give significantly better performance than existing implementations, and the absence of a VM allows simple programs to be very small indeed. MT claims to be able to make DLLs (which, to my knowledge, no existing Smalltalk can do), and to fit a trivial windows application into a 100K executable (their runtime-support DLL adds only 52K).

Dolphin Smalltalk is bytecode interpreted. I would expect this to offer significantly slower performance than current implementations, but with much lower space consumption, even for significant-sized applications. The company states that it "will be producing an optional JIT in future." When space is more important than speed, this is a very reasonable option, and this is often true in applets. The firm also says it will shortly have its VM encapsulated as an ActiveX downloadable.

SIZE

Both companies advertise their ability to make very small executables. Traditionally this has been a weak area for Smalltalk. In many of the application areas for which Smalltalk is used, one can argue that size doesn't matter as long as it does the job. But with the growing importance of small applets over monolithic applications, size is becoming more significant.

Dolphin's approach to minimizing size uses bytecoding and a very small interpreter (I counted 155K including all the DLLs) with a relatively standard (but small) image. Their prerelease packaging support wasn't very sophisticated, but when a development image starts at only 1.4 MB a good stripper is more of a luxury.

Smalltalk MT can provide extremely small sizes for simple programs, but for larger programs the space cost of full compilation becomes significant. The firm tries to combat this through modular class libraries and a minimalist framework.

The typical Smalltalk programmer doesn't think too

Alan Knight is a Smalltalk guru with The Object People. He can be reached by email as knight@acm.org, or at 613. 225. 8812.

much before using something that's in the development image. This is nice for development but bad for packaging. Everything depends on everything else, and all you can do is remove the compiler and development tools. MT claims to have kept the base classes small and modular, so if you don't use a particular class or subsystem it can easily be removed. They also claim to have a sophisticated packager, though I'm skeptical. Packaging is a difficult problem. I figure their base image, including the compiler and development DLLs, is 1.3 MB. "Image generation is largely automatic. An Image Builder computes the set of referenced classes and methods, starting with the image-entry point."

MT has also minimized the size of the runtime by sticking very close to the OS functionality.

"All GUI and operating system-related classes present a Win32-like protocol. This ensures that applications run with minimum overhead on the Windows family of operating systems, and leverage on existing know-how."

Smalltalk is a very loose term. While there are some clear, defining characteristics, there are also many variations as well.

FRAMEWORKS

A Win32-like protocol for windowing is definitely a two-edged feature. It makes Windows programmers who have programmed in other languages feel right at home, and it minimizes the amount of code you need to map a Smalltalk framework onto the operating system. I think there's no question that this reduces the amount of code in the image.

On the other hand, frameworks that are written with C or C++ in mind look *really* ugly compared to those designed for Smalltalk. Look at Visual Smalltalk vs. IBM. Visual Smalltalk has tight integration with Windows, to the point where you can implement your own wm:whatever: messages, but provides a very clean, simple, Smalltalk-like framework above that. IBM has its X/Motif layer, which translates into calls to the real OS. It works very well, and it's extremely portable, but it's way too close to C programming for my taste. IBM's saving grace is that with VisualAge or WindowBuilder you rarely have to descend to that level. Smalltalk MT doesn't appear to have a GUI builder. On the other hand, I'm sure there are lots of people using C/C++ for whom a Win32-like framework in Smalltalk (especially a Smalltalk that can produce fast, compact executables) would be a big step up.

Dolphin Smalltalk, in contrast, uses MVC. This is the oldest and best known of the Smalltalk frameworks, and has been passed down from PARC into VisualWorks. Dolphin has adapted it to a native-widget, event-driven form, but it still has Views, Controllers, and even ValueModels. They've also included a mediator class (called Tool) analogous to the Application Model of VisualWorks. I haven't tried building a window with either dialect but I'd expect to feel more comfortable with Dolphin.

THREADS

One of the most innovative features of Smalltalk MT is its full support for operating system threads ("true" multithreading). Making a decent compiler isn't that hard, but there are many issues involved in making Smalltalk use OS threads. The only other implementation I know of is OTI's embedded systems implementation, and it only supports threads for real-time operating systems.

I believe OS threads are often given too much importance. For 90% of applications, the standard Smalltalk process model will suffice, along with the ability to make system calls without blocking. The critical issue is that when one Smalltalk process waits on a system service, the entire Smalltalk system should not wait. Given this capability, which most implementations provide, it's not difficult to render demanding applications like Web or terminal servers, and you don't have to deal with the additional complexities of OS threads. With the Smalltalk model you can even have thread-manipulating code that's portable between operating systems.

That's a very good solution when it works, but there are times when you really need OS threads. One example is symmetric multiprocessing (SMP). Operating systems that support multiple CPUs in one machine can map OS threads to different processors, but not Smalltalk processes. Presently, this affects only very high-end machines, but it will become more and more important in the future.

Given that there are lots of implementations out there with the standard threading model, the choice of using OS threads in Smalltalk MT is a very welcome one indeed.

PLATFORMS AND PRICE

By now I 'll bet you are impatient to get your own copy of either or both of these implementations for your favorite platform. Unless you're a Microsoft fan you're out of luck. Both of these implementations have opted away from portability in favor of close integration with Win32. In fact, the first version of Smalltalk MT runs only on Windows NT 4.0 due to problems with Win95's threads, but they expect to have a Windows 95 version soon .

Dolphin Smalltalk has gotten more attention on the Net, because it can also run on older versions of NT and Windows 95. Don't expect them to expand their list of platforms too much. In response to a number of calls for an OS/2 version, Andy Bower of Dolphin's support group (Dolphin.support@intuitive.co.uk) wrote:

"The initial design aims for Dolphin were firmly directed to producing a great Win32 development system and this assumption is built into the product at a low level….we'd be reluctant to compromise this for compatibility with other environments."

As for pricing, both are very low compared to the current standards. Neither has fixed a firm price yet, but from what I've heard MT is approximately U.S. $300, and Dolphin at under U.S. $200.

continued on page 26

never really "get OO." After a reasonable time period (perhaps as long as nine months), the people who still haven't gotten it need to be given alternatives. Neither the mentor nor the developer is to blame. Not all people are able to think abstractly, and they need to be given the chance to contribute to the organization in a job for which they are suited.

CONCLUSION

"Smalltalk guru" is not the equivalent to Smalltalk mentor. Not all team members will accept mentoring, and not all team members will get OO. Do the best you can as a mentor and as a developer, and try to keep egos out of the equation. If personality clashes are a problem, maybe the mentor has to go. This is a tough call that the manager will have to make. Good luck and happy mentoring! S

References

1. THE WORDSWORTH CONCISE ENGLISH DICTIONARY, Hertfordshire. Wordsworth Editions Ltd., 1994.
2. Steinman, Jan, and Yates, Barabara, "Secrets to Building Successful Smalltalk Teams," tutorial at Smalltalk Solutions, March 1996, New York, N.Y.
3. Steinman, Jan, and Yates, Barbara, "Special" Team Members," *The Smalltalk Report*, V5N6, February 1996, pp. 15-17, 28.

continued from page 23

OVERALL

Overall I'm quite impressed with both of these implementations. As prerelease products, they're obviously immature in some areas. For example, I had difficulty with the debugger in both of them. The GUI builder in Dolphin didn't work yet, and MT doesn't appear to have one. On the other hand, in a lot of areas, they're surprisingly mature. They already have advanced features like finalization and exception handling in place. Inevitably, it will be a while before they're fully loaded with those features that have nothing to do with a language, and everything to do with a successful project: industrial-strength source code control, native database connections, extra widgetry, report writers, business graphics, and so forth. Nevertheless, they show enormous potential, and are well worth your while to investigate.

Of the two, I expect MT to be the first choice for those looking for cool new features, and for those doing things that are traditionally difficult in Smalltalk (e.g. server-based Smalltalk, very tiny apps). Restriction to the newest version of NT will lessen their impact in the short term. Dolphin Smalltalk, with a very low price, Win95 support, MVC, and ActiveX applet support, has real potential to become the Smalltalk for the masses.

Both are filling niches that are under-represented by current implementations, and I hope they will enjoy great success. S