Dave Thomas

Ubiquitous Applications: Embedded Systems to Mainframe

Over the last 10 years, Smalltalk has moved from the "Parc" to Main Street as a standard object-oriented (OO) fifth generation language (5GL) for enterprise computing. To meet the needs of application developers, Smalltalk environments and tools have matured from the original research implementations to full-featured, multiplatform development environments. A recent study of development tools conducted by Software Productivity Research in Massachusetts for a software productivity consortium ranked Smalltalk first in most categories. What is surprising about this study is the application: a demanding telephone switch traditionally dominated by C or proprietary talc languages such as Chill, Protel, and Plex. The fact that Smalltalk ranked so highly is a testimony that Smalltalk is an application 5GL that scales. This article discusses the major technical challenges addressed by Smalltalk implementors and application developers working on a wide spectrum of applications.

Multiplatform Portability of Applications and Objects

Smalltalk originally used proprietary Xerox workstations with integrated graphical displays. The early 1980s saw the development of efficient implementation techniques, including sophisticated method caches and generation-scavenging garbage collection, that allowed the implementation of compact interpreters for popular PCs and efficient 32-bit implementations for workstations. These techniques form the basis for modern commercial Smalltalk offerings, including VisualAge, Visual Smalltalk, and VisualWorks.

To support a variety of platforms, Smalltalk uses a virtual machine approach to portability that allows applications to be easily migrated between platforms with little or no change to the code. While application code portability was a problem Smalltalk solved early, application developers quickly encountered the need to persistently store and communicate

Smalltalk objects. Smalltalk implementors responded with object filing and swapping enhancements that allow objects, including cyclic ones, to be moved efficiently between images with transparent data translation.

The Smalltalk-80 MVC, which first appeared in the Apple Lisa personal computer, inspired many of the modern window and mouse-based interfaces. While Smalltalk originally provided the complete interface, recent implementations provide full support for multiplatform graphical user interfaces (GUIs) portability, including support for native widgets. Today's implementations include the ability to easily interoperate with other languages and platform capabilities via system application program interfaces. This allows the integration of Smalltalk and C to combine the best application language with the best systems programming language.

Source Code Control, Configuration Management, and Collaborative Development

Early development experiences supported by the Canadian Department of National Defense pointed out the need for team programming, configuration management, and development tools integrated with the Smalltalk development environment. This research effort led to the development of Orwell [5] and its commercial successor ENVY/Developer. These tools allow groups of developers to concurrently share fine-grained collections of classes known as applications or packages. Orwell introduced the concepts of component ownership, class extensions, and configuration maps to allow developers to work together in a shared, distributed image.

The ability to manage large class libraries across multiple development teams has been a critical success factor for many engineering and information technology organizations adopting Smalltalk. It is important to note the technology is only a contributing factor since the real challenge is making the 90-degree shift from a vertical functional structure to a horizontal component-based organization.

Enabling Client-Server Computing

During the past three years, Smalltalk has been augmented with tools for building graphical interfaces that allow the construction of portable GUIs. In order to make Smalltalk more accessible, visual programming tools, such as Fabrik, InterCONS, Parts, and VisualAge, allow end users to program by connecting visual components and operations.

In many applications, legacy requirements or online transaction processing (OLTP) performance requires support for existing relational and file systems. Additional class libraries for communications, transaction processing, and routine tasks such as report generation complete the suite of client/server tools. All commercial systems include database access frameworks that allow application programmers to access data from virtually any database. One of the design problems is how to efficiently, yet simply, map data between record/tuples and objects. Recent tools and frameworks, such as ObjectLens, have reduced the complexity for the application programmer, but as yet no products for high-performance transparent persistence interfaces compete against native access to static structured query language (SQL) or transaction processing (TP) facilities.

Clearly an OO database provides an alternative for storing Smalltalk objects, and vendors have responded to customer demand for Smalltalk clients by providing interfaces to their object servers. These are promising for new green-field applications and specialized applications such as multimedia or CAD.

Embedded Smalltalk: Can You Run It in a Watch?

Productivity has lead some developers to consider Smalltalk for embedded applications. These applications have limited RAM and ROM budgets, making them very different from workstation-based applications. During the process of embedding Smalltalk into Tektronix oscilloscopes (TDS 500 series) [2], we worked with Tektronix engineers to provide a series of tools for embedded system development and delivery. We customized the virtual machine to support separate RAM and ROM spaces for both code and data and provided tunable garbage collection, inlining of dynamic initialization code, and numerous space optimizations.

A tool called a packager allows the "ROMing engineer" to selectively place individual classes, methods, or instances into application/product-specific RAM and ROM. Packaging, unlike linking, which adds modules into a build process, removes unneeded development tools and components from the shared development environment. The Tektronix oscilloscopes have 512K of ROM and 32K of RAM. This is perhaps surprising for a language that is supposedly very large. The same technology was used to package DCS DbPublisher [4], a 512K PC-based Smalltalk/V application, into a 384K full-function database publishing product. The development tools [3] provide separate target and development images to support cross development.

The ability to manage large class libraries across multiple development teams *has been a critical success factor for many engineering and information technology organizations adopting Smalltalk.*

The use of a compact, customizable virtual machine technology allows the virtual machine to be tailored to the product, supporting specialized embedded resources such as cache, on-chip RAM/ROM, digital signal processors (DSPs), and low-power execution. This flexibility allows Texas Instruments' ControlWorks [1] to use Smalltalk in the embedded controllers as well as in application-level tools for wafer fabrication. Embedded applications need to be carefully designed and do not readily evolve from workstation prototypes with unconstrained resources. Specialized frameworks and considerable expertise are needed to deploy applications in constrained environments, such as personal digital assistants.

Mainframe Smalltalk: Can it Run Under CICS?

While multilevel client-server architectures seem to be the wave of the future, a large percentage of corporate data still lives in mainframes. For many applications it is much more efficient and simple to run portions of the Smalltalk application on the host platform where the methods can run efficiently against the data. Much to the surprise of many, IBM has just introduced MVS Smalltalk, which allows server portions of a distributed Smalltalk application to execute on a CICS/DB2 or IMS mainframe.

The substantial technical challenges of developing MVS Smalltalk include running cross development, packaging small transaction program images, executing multiple transactions against the same transaction application, and integrating with host resources such as transaction monitors and databases. Its availability means Smalltalk can be used across all platforms in the business enterprise. This allows the application designer to place portions of the application in the most appropriate processor for security and performance.

Distributed Smalltalk and Server Smalltalk

The message-oriented nature of Smalltalk makes it a natural for concurrent and distributed object computing. However, the symbolic nature of Smalltalk relies heavily on a single, shared object space, and such issues as safe multithreading and concurrent/distributed garbage collection need to be addressed to build large-scale applications. Despite several early research projects, industrial offerings have only recently begun to appear. Hewlett-Packard was the first to introduce a distributed Smalltalk compliant with the Object Management Group (OMG) object request broker (ORB) specification followed closely by IBM with a distributed system object model (DSOM) implementation. These activities led to the definition and adoption of an OMG-approved Smalltalk ORB binding. IBM recently demonstrated a distributed Smalltalk based on a uniform address space.

Several research projects including Actra and Actalk have investigated concurrent Smalltalk. With the widespread availability of symmetric multiprocessing (SMP) servers, it is likely we will see true multithreaded, multiprocessor implementations in the near future. Research on massively parallel processor (MPP) systems, as exemplified by Coda, explores how Smalltalk can be scaled to massively parallel systems. Both shared object space and communicating object space are likely to be supported since each offers advantages for specific applications.

These implementations have been used successfully in pilot and prototype industrial projects, but more experience is required to determine the programming models, tools, and transaction services needed to support the development of mission critical distributed multithreaded applications.

Conclusion

Smalltalk has succeeded where other high-productivity environments such as Lisp, APL, Forth, Prolog, and to some extent Basic have failed. These languages promised interactive, incremental development in a supporting environment, in some cases with libraries of tools and components. Unlike popular proprietary 4GLs, Smalltalk is a standard 5GL that scales. Critical success factors for Smalltalk are portability, collaborative development, packaging, standard class libraries, and scaling from single user to distributed applications.

While challenges remain in embedded, distributed, and server Smalltalk research and practical experience suggest that these areas will be addressed in the near future. Given continued research and development, Smalltalk applications can be deployed from a mainframe to a wrist watch. This makes Smalltalk a ubiquitous application development environment.

References

1. Barry B. Smalltalk as a Development Environment for Integrated Manufacturing Systems. In Proceedings of the International Conference on Object-Oriented Manufacturing Systems. Division of Manufacturing Engineering, Univ. of Calgary, 1992.
2. Dotts, A., Birley, D. Developer of reusable test equipment software using Smalltalk. In *Proceeedings of OOPSLA 92*. Addendum to the Proceedings, *OOPS Mess. 4*, 2 (Apr. 1993), pp. 31–36.
3. Duimovich, J. Milinkovich, M. Smalltalk and embedded Systems. *Dr. Dobb's J. 181* (Oct. 1991), 86–95.
4. Thomas, D., Best, R. Dbpublisher—Tex in shining armor: An integrated system combining Tex and Smalltalk. Woodman 89: Workshop on Object-Oriented Document Manipulation, Rennes, France (May 29–31, 1989), pp. 280-285.
5. Thomas, D., Johnson, K. Orwell: A configuration management system for team programming. Special issue of *Sigplan Not. 23*, 11 (Nov. 1988), 135–141.

David Thomas *is founder and CEO of Object Technology International, Inc. (OTI), Ontario, Canada.*