Programming as an Experience: The Inspiration for Self

Randall B. Smith and David Ungar

Introduction by David Ungar

The Self project started in 1986 as a language design exercise at Xerox PARC. Randy Smith and I (with a germ of inspiration from Peter Deutsch) tried to come up with a language design that, without implementation compromises, would give the programmer even more of the benefits of object−oriented programming than Smalltalk (our then−favorite) did. To that end, we sought simplicity by unifying disparate concepts and were pleased and surprised to discover that the simplification enhanced rather than diminished the language's expressive power. This experience was so significant to us that we chose to call the first Self paper: "Self: The Power of Simplicity."

Randy went off to England, and I went off in search of a research area for my graduate students and myself at Stanford. Taking a leap of faith, I decided to see if it were actually possible to turn this paper design into a full−fledged programming environment, with a speedy implementation and rich graphical user interface. Craig Chambers, Elgin Lee, Bay−Wei Chang, Urs Hoelzle, and Ole Agesen joined me at Stanford to pursue this vision. Emil Sarpa, Wayne Rosing, and Bill Joy at Sun also became interested in our project and, along with other corporations and government grants, helped provide the early funding.

We were able to achieve some interesting results in the areas of object−oriented high−performance implementation techniques and the application of cartoon animation to enhance the legibility of graphical user interfaces. In 1991, the project left Stanford to join the fledgling Sun Microsystems Laboratories. I was delighted when Randy Smith decided to leave PARC and join us as co−leader of the project here at Sun. And Jim Mitchell, now head of Sun Labs, lent us his management expertise, helping us settle on a specific roadmap and schedule for creating the full Self environment. Over the next two years, three more talented individuals, Lars Bak, John Maloney, and Mario Wolczko joined us at Sun.

Almost three years later, we had reached our goal of developing a complete system with many novel characteristics, only to be dealt a difficult blow. Feeling that we would not be able to attract a significant user community, Sun Microsystems Laboratories decided to cancel the project. Since then, the members of the Self group have gone on to contribute to object−oriented systems both inside and outside of Sun:

− Craig Chambers and Urs Hoelzle, two Ph.D. students who were sponsored by Sun, have had successful academic careers and made their own contributions to the field of object−oriented languages and implementation techniques.

− Bay−Wei Chang went to Xerox PARC to work on graphical interfaces.

− John Maloney left Sun to join Alan Kay at Apple, where he was instrumental in implementing Squeak, a publicly−available Smalltalk system. Later John reimplemented the Morphic GUI framework that he and Lars had built under Randy's direction for Self.

− Lars Bak and Urs Hoelzle joined a startup, which was later acquired by Sun and built the Java HotSpot™ Java™ Virtual Machine, the core of Sun's desktop and server Java platform.

− Ole Agesen went on to start and drive the ExactVM project, Sun's first JVM™ with accurate garbage collection. (See paper #15 − The GC Interface in the EVM.)

− Mario Wolczko became West Coast tech lead for the ExactVM project for the first six months of the project, manager of the Spotless (Java™ Virtual Machine for small devices) project during its transition to product land, and has recently led the design of the hardware performance instrumentation architecture in a design for a future SPARC™ processor.

− Randy Smith exploited the Self system as a basis for several other projects in the Labs, including the Distributed Tutored Video Instruction work, (See paper #17 − Experiments Comparing Face−to−Face with Virtual Collaborative Learning). The work done in Self also influenced the "Nebraska" project and the JAMM (Java Applets Made Multi−User) Project.

− The K Virtual Machine (KVM) Java implementation, ubiquitous on cell phones today, is based on the Sun Labs Spotless project, undertaken by Antero Taivalsaari and Bill Bush. Antero and Bill were initially hired to participate in a successor to the Self project.

− Finally, David Ungar was loaned out to JavaSoft™ from Sun Labs. There he built the portability framework for Java HotSpot™ and built the first version of HotSpot to run on SPARC™ instead of Intel processors. (David just ported the interpreter−−others built the compilers for SPARC.)

The various facets of Self have had varying amounts of impact. NewtonScript and other, lesser−known languages were heavily influenced by the language. The Morphic GUI framework has been used by many in the Squeak community. But the largest impact has been in the area of implementation techniques.

Perhaps because we were forced to think ambitiously in order to make Self practical, we found techniques, such as transparent, adaptive optimization, that have been harnessed for Java and many other subsequent projects. Descendants of Self's implementation techniques have aided Java's acceptance by slashing the performance penalty faced by Java's adopters.

All of Sun's Java Virtual Machines since the original have benefited from key participation by Self alumni or extended alumni (such as Bill Bush, Antero Taivalsaari, and Phillip Yelland).

REFERENCES:

Retrospective:

Programming as an Experience: The Inspiration for Self (1995) Randall B. Smith and David Ungar. Invited paper for ECOOP'95. Reprinted in "Prototype−Based Programming: Concepts, Languages and Applications," James Noble, I. Moore, A. Taivalsaari, eds. Springer−Verlag.

Language:

Self: The Power of Simplicity (1987) David Ungar and Randall B. Smith. OOPLSA'87. Revised version published in Journal of Lisp and Symbolic Computation, 4(3), Kluwer Academic Publishers, June, 1991. TR−94−30

Parents are Shared Parts: Inheritance and Encapsulation in Self (1991) Craig Chambers, David Ungar, Bay−Wei Chang, and Urs Hoelzle. Journal of Lisp and Symbolic Computation, 4(3), Kluwer Academic Publishers, June, 1991.

Organizing Programs Without Classes (1991) David Ungar, Craig Chambers, Bay−Wei Chang, and Urs Hoelzle. Journal of Lisp and Symbolic Computation, 4(3), Kluwer Academic Publishers, June, 1991.

Implementation:

Object Storage and Inheritance for Self (1988) Elgin Lee. Stanford University Engineer's thesis.

Customization: Optimizing Compiler Technology for Self, a Dynamically−Typed Object−Oriented Programming Language (1989) Craig Chambers and David Ungar. PLDI'89.

An Efficient Implementation of Self, a Dynamically−Typed Object−Oriented Language Based on Prototypes (1989) Craig Chambers, David Ungar, and Elgin Lee. OOPSLA'89. Also in the Journal of Lisp and Symbolic Computation, 4(3), Kluwer Academic Publishers, June, 1991.

Iterative Type Analysis and Extended Message Splitting: Optimizing Dynamically−Typed Object−Oriented Programs (1990) Craig Chambers and David Ungar. PLDI'90. Also in the Journal of Lisp and Symbolic Computation, 4(3), Kluwer Academic Publishers, June, 1991.

Making Pure Object−Oriented Languages Practical (1991) Craig Chambers and David Ungar. OOPSLA'91.

Optimizing Dynamically−Typed Object−Oriented Programming Languages with Polymorphic Inline Caches (1991) Urs Hoelzle, Craig Chambers, and David Ungar. ECOOP'91.

The Design and Implementation of the Self Compiler, an Optimizing Compiler for Object−Oriented Programming Languages (1992) Craig Chambers. Stanford Doctoral Dissertation.

Debugging Optimized Code with Dynamic Deoptimization (1992) Urs Hoelzle, Craig Chambers, and David Ungar. OOPSLA'92.

Object, Message, and Performance: How They Coexist in Self (1992) David Ungar, Randall B. Smith, Craig Chambers, and Urs Hoelzle. I.E.E.E. Computer Magazine, October, 1992. Reprinted in Readings in Object−Oriented Systems and Applications, David Rine, ed, IEEE Computer Society Press.

Fast Write Barrier for Generational Garbage Collection," U. Hoelzle, OOPSLA'93.

Optimizing Dynamically−Dispatched Calls with Run−Time Type Feedback (1994) Urs Hoelzle and David Ungar. PLDI'94.

Adaptive optimization for Self: Reconciling High Performance with Exploratory Programming (1994) Urs Hoelzle. Stanford doctoral dissertation.

A Third−Generation Self Implementation, Urs Hoelzle and David Ungar. OOPSLA '94.

Do object−oriented languages need special hardware support? Urs Hoelzle and David Ungar. ECOOP '95.

Adaptive Optimization for Self: Reconciling High Performance with Exploratory Programming, Urs Hoelzle, 1994 Stanford doctoral dissertation. TR−95−35

The Design and Implementation of the Self Compiler, an Optimizing Compiler for Object−Oriented Programming Languages, Craig Chambers. 1992 Stanford doctoral dissertation.

User Interface:

Experiencing Self Objects: An Object−Based Artificial Reality (1990) Bay−Wei Chang and David Ungar.

The Use−Mention Perspective on Programming for the Interface (1992) Randall B. Smith, David Ungar, and Bay−Wei Chang. In Computer Languages for Programming User Interface Software, a workshop at the ACM SIGCHI'91 conference, Brad A. Myers, organizer and editor. Reprinted in Languages for Developing User Interfaces, Brad Myers, ed., Jones and Bartlett, London, 1992.

Animation: From Cartoons to the User Interface (1993) Bay−Wei Chang and David Ungar. UIST'93. TR 94−33

Getting Close to Objects: Object−Focused Programming Environments (1995) Bay−Wei Chang, David Ungar, and Randall B. Smith. In Visual Object Oriented Programming, Margaret Burnett, Adele Goldberg, & Ted Lewis, eds., Prentice−Hall, 1995.

The Self−4.0 User Interface: Manifesting a System−wide Vision of Concreteness, Uniformity, and Flexibility (1995) Randall B. Smith, John Maloney, and David Ungar. OOPSLA'95.

Concreteness and Cartoon Animation in Seity, A user Interface for Object−Oriented Programming, Bay−Wei Chang. 1996 Stanford Doctoral Dissertation.

Type Inference and Application Extraction:

Type Inference of Self: Analysis of Objects with Dynamic and Multiple Inheritance (ECOOP'93) Ole Agesen, Jens Palsberg, and Michael I. Schwartzbach

Constraint−Based Type Inference and Parametric Polymorphism (SAS'94) Ole Agesen

Sifting Out the Gold: Delivering Compact Applications From an Exploratory Object−Oriented Programming Environment (OOPSLA'94) Ole Agesen and David Ungar.

The Cartesian Product Algorithm (ECOOP'95) Ole Agesen.

Type Feedback vs. Type Inference: A Comparison of Optimization Techniques for Object−Oriented Languages (OOPSLA'95) Ole Agesen and Urs Hoelzle.

The Design and Application of Type Inference for Object Oriented Programming Environments, Ole Agesen. I995 Stanford doctoral dissertation. TR 96−52

Other:

Integrating Independently−Developed Components in Object−Oriented Languages (1993) Urs Hoelzle

Mango − A Parser Generator for Self (1994) Ole Agesen TR 94−27