Panel

Celebrating 40 Years of Language Evolution: Simula 67 to the Present and Beyond

Steven Fraser
Director (Engineering)
Cisco Research Center
Cisco Systems, San Jose

James Gosling
VP and Sun Fellow
Sun Microsystems
Menlo Park

Anders Hejlsberg
Technical Fellow
Microsoft
Redmond

Ole Lehrman Madsen
Professor
Computer Science
Aarhus University

Bertrand Meyer
Professor and Advisor
ETH Zurich

Guy Steele
Sun Fellow
Sun Microsystems
Burlington

Abstract

Simula 67 (SIMple Universal LAnguage 67) is considered by many as one of the earliest – if not the first – object-oriented language. Simula 67 was developed by Ole-Johan Dahl and Kristen Nygaard in Oslo, Norway and has greatly influenced object-oriented language development over the past 40 years. This panel brings together leading programming language innovators to discuss and debate past, present, and future language evolutions.

Categories & Subject Descriptors:
D.3 Programming Languages
K.0 Computing Milieux
K.4.3 Organizational Impacts

General Terms: Languages

Keywords: Simula67, Java, C#, BETA, Eiffel, Scheme, C

1. Steven Fraser *(panel impresario)*, sdfraser@acm.org

STEVEN FRASER recently joined the Cisco Research Center in San Jose California as a Director (Engineering) with responsibilities for developing and managing university research collaborations. Previously, Steven was a Senior Staff member of Qualcomm's Learning Center in San Diego leading software technical learning and development programs and creating the corporation's internal technical conference – the *QTech Forum*. Steven also held a variety of technology management roles at Nortel/NT/BNR including: Process Architect, Senior Manager (Disruptive Technology and Global External Research), and Design Process Engineering Advisor. In 1994 he spent a year as a Visiting Scientist at the Software Engineering Institute (SEI) collaborating with the Application of Software Models project on the development of team-based domain analysis (software reuse) techniques. Fraser is the Corporate Support Chair for OOPSLA'07 and was the General Chair for XP2006. Fraser holds a doctorate in EE from McGill University in Montréal – and is a member of the ACM and a senior member of the IEEE.

OOPSLA'07 October 21–25, 2007, Montréal, Québec, Canada.
ACM 978-1-59593-865-7/07/0010.

2. James Gosling, *James.Gosling@sun.com*

JAMES GOSLING received a B.Sc. in Computer Science from the University of Calgary, Canada in 1977. He received a Ph.D. in Computer Science from Carnegie-Mellon University in 1983. The title of his thesis was *The Algebraic Manipulation of Constraints*. He has built satellite data acquisition systems, a multiprocessor version of Unix, several compilers, mail systems and window managers. He has also built a WYSIWYG text editor, a constraint based drawing editor and a text editor called 'Emacs' for Unix systems. At Sun his early activity was as lead engineer of the NeWS window system. He did the original design of the Java programming language and implemented its original compiler and virtual machine. In February 2007, James was named an officer of the Order of Canada.

It's tempting, at a conference like OOPSLA, to focus on the design of programming languages. Languages arise from a context. They are driven by the needs of developers, the problems they are trying to solve, and the systems they are trying to put together. Simula 67 was driven by simulation. Java was driven by networking. What are the trends that will drive the next generation of languages? High on my list of forces is the march of Moore's law and its transition from increasing clock rates to increasing core counts and specialized cores (will programming GPUs ever become mainstream?). Moore's law will also push computing over some threshold's where some interesting problems crumble: for example, will programming change if speech recognition and AI become more generally usable? The problem domains also drive languages. We currently have a generation of web languages optimized for rapid flexible development. On the other hand, the scale of software that deals with "real reality" (as opposed to "virtual reality") is exploding, and here, testability and reliability are far more important than RAD.

3. Anders Hejlsberg, andersh@microsoft.com

ANDERS HEJLSBERG is a Technical Fellow in the Developer Division at Microsoft. Hejlsberg is recognized as an influential creator of development tools and programming languages. He is the chief designer of the C# programming language and a key participant in the development of the Microsoft .NET Framework. Before his work on C# and the .NET Framework, Hejlsberg was an architect for the Visual J++ development system and the Windows Foundation Classes. Before joining Microsoft in 1996, Hejlsberg was one of the first employees of Borland International Inc. As principal engineer, he was the original author of Turbo Pascal, a revolutionary integrated development environment, and chief architect of its successor, Delphi. Hejlsberg co-authored *The C# Programming Language*, published by Addison Wesley, and has received numerous software patents. In 2001, Hejlsberg was the recipient of the prestigious Dr. Dobbs Excellence in Programming Award. Hejlsberg studied engineering at the Technical University of Denmark.

Thoughts about the future:
The next generation of programming languages will fuse ideas from dynamic, static, functional, and domain specific languages. We are already seeing the beginnings of this today: Dynamic languages are adopting static typing, static languages are incorporating implicit typing, concepts from functional languages are seeing mainstream adoption, and domain specific languages such as HTML, SQL, XAML, and XSLT are an integral part of today's applications.

For all of their advances, today's mainstream languages are very imperative in nature. Sequential statements with explicit control flow and mutation of state lead to over-specification of solutions and algorithms. Programs say not just *what* they want done, but also, in excruciating detail, *how* they want it done. This level of detail makes programming tedious and error prone. Future progress and innovation in programming languages will increasingly occur in the area of declarative programming where you say the "what" but not the "how", simply because this style of programming leaves more room for intelligence in the execution infrastructure. The LINQ (Language INtegrated Query) capabilities introduced in C# 3.0 are an example of this trend.

Separately, we need to create a better programming model for concurrent programs. Concurrency is the new reality. Going forward, computers will have more CPUs rather than faster CPUs. This is a fact. It is also a fact that existing concurrent programming models are too complicated for all but the top echelon of programmers to master. Interestingly, successful concurrent/parallel programming models characteristically "hide" the concurrency and present a simpler serial, transactional, and/or functional programming paradigm.

4. Ole Lehrmann Madsen, ole.l.madsen@alexandra.dk

OLE LEHRMANN MADSEN (www.daimi.au.dk/~olm) is a professor of Computer Science, Aarhus University, and director of the Alexandra Institute A/S (www.alexandra.dk) – a joint venture between universities, companies and public institutions to promote private and public co-operation within IT research. He is a co-founder and chairman of the board for Mjølner Informatics (www.mjolner.com). He has worked with object-technology for more than 25 years starting with SIMULA programming. He developed the BETA programming language together with Kristen Nygaard, Birger Møller-Pedersen and Bent Bruun Kristensen, and he has been a research manager for the Mjølner project where the first version of the BETA software was developed. An important part of the BETA project was the development of a conceptual framework for object-oriented programming. The conceptual framework consists of conceptual means such as abstraction, classification and composition for understanding and organizing knowledge about the real world. BETA was presented at the ACM conference on History of Programming Languages III in San Diego, June 2007.

Language mechanisms like class, subclass, and virtual method originating from SIMULA are found in most mainstream object-oriented languages. In addition a number of new mechanisms such as generics, meta classes, and multiple inheritance have been proposed. The research literature contains numerous suggestions for more or less interesting new language constructs. In my opinion the main challenges for language designers are within the following areas:

- We need better language mechanisms for concurrent and distributed programming, especially in the context of pervasive and/or ubiquitous computing
- We need a new form of abstraction mechanisms that also cover design patterns
- We need an integration of class-based and prototype-based languages

We need to reintroduce the importance of a programming language to be useful for modeling as was the case with SIMULA.

5. Bertrand Meyer, Bertrand.Meyer@inf.ethz.ch

BERTRAND MEYER is a professor of software engineering at ETH Zurich and a scientific advisor of Eiffel Software in California. He is the developer of the Eiffel method and language, now an ISO standard, and originator of the method *Design by Contract*. He has published nine books translated into a dozen languages, including "Object-Oriented Software Construction" (Jolt Award 1997). He has directed the design and implementation of numerous tools and libraries used in mission-critical applications and is active on both the academic and industrial scenes as a speaker and consultant; his scientific contributions include more than 200 scholarly articles. He is among other awards the recipient of the first Dahl-Nygaard award for object technology (2005) and the ACM Software System award (2007).

Eiffel is rooted in two technologies: Simula 67, which exerted the most directed influence since Eiffel was designed from the experience of Simula use over many years; and the methods and languages of formal specification, proof and verification. To this basis Eiffel has over its twenty-year history added considerable enhancements, always with a view to preserving the coherence and simplicity of the edifice. Many concepts now considered standard in OO languages were pioneered by Eiffel, from genericity and the combination of static typing and dynamic binding to efficient garbage collection and closure-like mechanisms (agents, delegates) bringing over techniques similar to those of functional languages. Others present in Eiffel for a long time, such as Design by Contract constructs, are making their way into research languages. Yet others (such as covariance), used in Eiffel as a matter of course, are still not widely accepted elsewhere. In addition, novel constructs and concepts continue to be explored and are progressively integrated into newer versions of the language.

6. Guy Steele, Guy.Steele@sun.com

GUY L. STEELE JR. (PhD, MIT, 1980) is a Sun Fellow at Sun Microsystems, Inc. In 1975 he co-invented the Scheme programming language at MIT with Prof. Gerald Jay Sussman. He is author or co-author of four books on programming languages (Common Lisp, C, High Performance Fortran, and the Java programming language). He is an ACM Fellow and received the 1996 ACM SIGPLAN Programming Languages Achievement Award. He designed the original EMACS command set and was the first person to port TeX. At Sun Microsystems he is responsible for research in language design and implementation strategies, and architectural and software support.

Language evolution has made some progress, but for every lesson learned, that lesson gets reinvented or rediscovered three or four times – which is to say that the lesson is forgotten three or four times. Maybe there's just too much for any one language designer or implementer to keep in his head all at once. Maybe progress is merely an illusion we layer over the historical sequence of fads that results as we focus our attention on one detail or another. People nod their heads when we say that Algol 60 and Simula 67 and Standard ML are classic, beautiful, well-thought-out designs, but a lot of the world's actual work gets done with C and Javascript and Perl.