Call for Papers: IFIP Working Conference on Domain Specific Languages
July 15-17, 2009, Oxford

http://www.hope.cs.rice.edu/twiki/bin/view/WG211/DSLWC

Domain-specific languages are emerging as a fundamental component of software engineering practice. DSLs are often introduced when new domains such as web-scripting or markup come into existence, but it is also common to see DSLs being introduced and adopted for traditional domains such as parsing and data description. Developing software using DSLs has many benefits. DSLs are often designed based on existing notations that are already in use by experts in a given domain. As such, successful DSLs often reduce or eliminate the effort needed to transform the concept or innovation produced by the domain expert into an executable artifact or even a deliverable software product. DSL implementations can capture and mechanize a significant portion of the repetitive and mechanical tasks that a domain expert traditionally needed to perform in order to produce an executable. DSLs can in many cases capture and make widely available special expertise that only top specialists in a given domain might have. By capturing expert knowledge and reducing repetitive tasks, DSLs often also lead to software that is significantly more portable, more reliable and more understandable than it would otherwise be.

DSLs can be viewed as having a dual role to general-purpose languages: whereas general purpose languages try to do everything as well as possible, DSLs are designed to find a domain where they can solve some class of problems – no matter how small – in the best possible way. Widely known examples of DSLs include Matlab, Verilog, SQL, LINQ, JavaScript, PERL, HTML, Open GL, Tcl/Tk, Macromedia Director, Mathematica/Maple, AutoLisp/AutoCAD, XSLT, RPM, Make, lex/yacc, LaTeX, PostScript, Excel, among many others. But while these tools have been widely successful, they still fall short of realizing the full ide3 behind them. The goal of this conference is to explore the extent to which incorporating modern principles of language design and software engineering can benefit existing and future domain-specific languages.

The ultimate goal of using DSLs is to improve programmer productivity and software quality. Often, this is achieved by reducing the cost of initial software development as well as maintenance costs. These improvements – programs being easier to write and maintain – materialize as a result of domain-specific guarantees, analyses, testing techniques, verification techniques, and optimizations.

Paper Criteria

Papers are sought addressing the research problems, fundamental principles, and practical techniques of DSLs, including but not limited to:

- Foundations, including semantics, formal methods, type theory, and complexity theory
- Language design, ranging from concrete syntax to semantic and typing issues
- Software engineering, including domain analysis, software design, and round-trip engineering
- Software processes, including metrics for software and language evaluation
- Implementation techniques, including parsing, compiling, and program generation
- Program analysis and automated transformation
- Reverse engineering, re-engineering, design discovery, automated refactoring
- Hardware/software codesign
- Programming environments, including visual languages, debuggers, and testing infrastructure
- Teaching DSLs and the use of DSLs in teaching

Case studies, including engineering, bioinformatics, hardware specification languages, parallel computing languages, real-time and embedded systems, and networked and distributed domains Papers will be judged on the depth of their insight and the extent to which they translate specific experience into general lessons for domain-specific language designers and implementers, and software engineers. Papers can range from the practical to the theoretical; where appropriate, they should refer to actual languages, tools, and techniques, provide pointers to full definitions and implementations, and include empirical data on results.

Important Dates

November 12th, 2008: Final Call for Papers
December 14th, 2008: Abstract submission due.
December 21st, 2008: Paper submission deadline.
February 23rd, 2009: Author notification of decisions
March 22nd, 2009: Camera ready manuscripts due

All deadlines are firm; no extensions will be given.

Organizers

General Chair: Jeremy Gibbons, Oxford University; **Publicity Chair:** Emir Pasalic, LogicBlox; **Program Committee**: Walid Taha (Chair), Rice University; Jon Bentley, Avayalabs; Martin Erwig, Oregon State University; Jeff Gray, University of Alabama at Birmingham; Robert Grimm, New York University; Jim Grundy, Intel Strategic CAD Labs; Tom Henzinger, EPFL; Sam Kamin, UIUC; Dick Kieburtz, Portland State University; Ralf Laemmel, University of Koblenz; Julia Lawall, University of Copenhagen; Benjamin Pierce, University of Pennsylvania; Vivek Sarkar, Rice University; Mary Sheeran, Chalmers University; Jeremy Siek, University of Colorado at Boulder; José Nuno Oliviera, University of Minho; Doaitse Swierstra, Utrecht University; Eelco Visser, Delft University; William Waite, University of Colorado at Boulder; Stephanie Weirich, University of Pennsylvania