Industrial Panel
at the DSL Dagstuhl
3 Feb 2015

Ralf Lämmel (Moderator)
University of Koblenz-Landau
Software Languages Team
rlaemmel@gmail.com

Shortened specification provided by the organizers

- **2 minutes 1 slide per panelist**
- Thoughts on these questions:
  - Do the academic approaches to DSL development work for industry?
  - What does industry need to get useful DSLs?
  - What is the actual practice for DSL development in your industrial setting?
- **Opening remark: identify the single most important issue/problem** that should be addressed (which might very well not be among those questions above).
- Synthesize from the individual positions the one most important issue regarding DSLs in industry.

Panelists

- ☐ Hassan Chafi, Oracle
- ☐ Steven Kelly, MetaCase
- ☐ Oleg Kiselyov, Tohoku University
- ☐ Julia Rubin, MIT
- ☐ Markus Voelter, independent/itemis

Hassan Chafi
Director, Research & Advanced Development @ Oracle

What are successful lifecycles for DSLs and their development?

How can DSLs be gradually introduced into existing projects (lots of valuable code already exists and would benefit from DSLs)?

What are interoperability models between DSLs and general purpose languages that either host them or glue them to other parts of the system?

Steven Kelly
MetaCase

Moving from characters to objects

- Don’t rebuild understanding from scratch, store the understood structure.
- Manipulate, co-operate, link on that level, not as characters.
- Allow free representation and persistent layout, not forced automatic layout
- Together, these allow scalability in model and team size and complexity

Good graphical DSLs shown to be 5-10x faster than programming

- 2-3x because it is a DSL (same as character-based textual DSL)
- 2-3x because it operates on structured objects with free layout
- Graphical vs. textual a relatively small factor, maybe 1-2x
- Graphical DSL widens potential effective user population by an order of magnitude

Oleg Kiselyov
Tohoku University

What exactly is a DSL?

The issue came up at the Shonan seminar which we organized past May. We were talking about some DSL for HPC, and then someone asked: why do we call it DSL? What exactly is „domain-specific“ here? What is the domain? HPC is so broad that the classification is almost useless. The issue is especially blurred for embedded DSLs. How is that different from a library?

Thus, who are the expected users and can they be expected to use the DSL, whatever it is, without the developers’ help?

Julia Rubin
MIT

- How to keep the language current

  when new requirements emerge

  when new technologies are introduced

- How to combine domain specific languages

- How to keep the language attractive

  e.g., to new employees that are unfamiliar with the language

Markus Voelter
independent/itemis

Scalability:

* **Scaling Languages, Domains and Audiences: wider ranges of notations and "language experiences", more stripped down, user-friendly tools**
* Scaling languages and language ecosystems in terms of complexity: more declarative and analyzable language specifications, fewer specs for more language aspects
* Scaling some aspects of tools: incremental generation for big systems, incremental type checking for non-trivial type system rules, ...