DiverSE’s Seminar about Software Language Engineering

May 28th, 2015
Rennes, France

THE DIVERSE TEAM
Software intensive systems
Software intensive systems

Multiple dimensions of software diversity
DiverSE

- Functional diversity
- Language diversity
- Execution diversity
- Failure diversity

Model driven engineering
DiverSE

Functional diversity

Language diversity

Model-driven engineering

Execution diversity

Failure diversity
Global scientific objective

• Automatically compose and synthesize software diversity from design to runtime to address unpredictable evolutions of software intensive systems
Scientific foundations

• Automated reasoning
  • logic and ontologic modeling and reasoning

• Metamodeling and language engineering
  • semantic specification, type theory, software language engineering

• Adaptive systems
  • distributed, component based systems and search-based algorithms

• Program analysis
  • program transformation, software testing, software diversity

• Empirical software engineering
Software development

• DiverSE’s research is experimental
  • all research results rely on the development of software tools and experiments

• Familiar
  • an environment for variability modeling
  • variability analysis, product derivation and variability reverse engineering

• Kevoree
  • development and deployment of component-based distributed software
  • dynamic adaptive systems

• Mélange
  • a modeling language workbench
  • MDE in the large (model slicing, composition, simulation, etc.)
DiverSE

• DIVERSE

• 8 faculty members

• ~ 30 PhD students, postdocs, and engineers
FROM MODEL (DRIVEN) ENGINEERING…
… TO LANGUAGE (DRIVEN) ENGINEERING
Applications Domains

Software intensive systems
Multiple Concerns

- Avionics
- Aerodynamics
- Propulsion System
- Communications
- Mechanical Structure
- Airlines
- Human-Machine Interaction
- Environmental Impact
- Safety Regulations
- Authorities
- Navigation
Change one Aspect and
Automatically Re-Weave:
From Software Product Lines...
..to Dynamically Adaptive Systems
"Perhaps surprisingly, the majority of MDE examples in our study followed domain-specific modeling paradigms"

Domain-Specific Languages (DSLs)

- Targeted to a **particular** kind of problem, with dedicated notations (textual or graphical), support (editor, checkers, etc.)
- Promises: more « efficient » languages for resolving a set of specific problems in a domain
Another lesson we should have learned from the recent past is that the development of 'richer' or 'more powerful' programming languages was a mistake in the sense that these baroque monstrosities, these conglomerations of idiosyncrasies, are really unmanageable, both mechanically and mentally.

aka **General-Purpose Languages**

I see a great future for very systematic and very modest programming languages

aka **Domain-Specific Languages**

1972

ACM Turing Lecture, « The Humble Programmer »
Edsger W. Dijkstra
« Domain-specific languages are far more prevalent than anticipated »
"Software languages are software too"

Software Language Engineering (SLE)

• Application of systematic, disciplined, and measurable approaches to the development, deployment, use, and maintenance of software languages

• Supported by various kind of "language workbench"
  • Eclipse EMF, xText, Sirius, GEMOC, Papyrus
  • Jetbrain’s MPS
  • MS DSL Tools
  • Etc.

• Various shapes and ways to implement software languages
  • External, internal or embedded DSLs, Profile, etc.
  • Grammar, metamodel, ontology, etc.

• More and more literature, a dedicated Intl. conference (ACM SLE, cf. http://www.sleconf.org)…
The Kermeta Workbench (since 2005)

The Kermeta Workbench (since 2005)

- **Modular design** of DSMLs
  - One meta-language per language concern (merge/weave)
    - Ecore, OCL, Xtend
    - But also: QVTo, fUML, Alf, Ket, Xsd…
  - Static introduction mechanism (aspect)

- Provides a **model oriented action language** to support common model manipulation tasks
  - to implement (E)Operation’s bodies
  - Imperative, statically typed, object-oriented, aspect-oriented (aspect/context, require), model-oriented, DbC, Unit testing
  - Java and Xtend compliant, and based on EMF
  - Run as Eclipse plugin or as standard Java application

- **Efficient implementation** of DSMLs
  - Mashup of the meta-languages to Java code
  - Integrated with third-party tools (EMF compliant)

TOWARDS LANGUAGE-ORIENTED MODELING*
FROM DESIGN TO RUNTIME

"A clear challenge, then, is how to integrate multiple DSLs."

"Rather than attempting to formalize a wide-ranging domain (such as financial applications), practitioners should write small, easy-to-maintain DSLs and code generators. In practice, however, multiple DSLs are usually required, which brings its own challenges in terms of integration."

Software Language Engineering (SLE)

All about “family of languages”!! 😊

Current activities related to SLE in DiverSE

DSL Design and Implementation
• Modularity, reuse, variability management and domain-specific metalanguages
• Executability and trace management
• Adaptability

DSL Integration
• Language interface (structural and behavioral)
• Language composition
• Language globalization

Application on (smart) CPS
In various projects: VaryMDE, DGA, GEMOC, MBSAR, MERgE, Clarity
CLOSED WORLD

Variability model

Language derivation

Variability-based development model for DSLs
- Variability modeling
- Components-based languages development

Families of Languages

Variants

Typing Theory for Agile Modeling
- Language interfaces
- Model polymorphism
- Viewpoints management

OPEN WORLD

Language Manipulation
- Evolution
- Extension
- Restriction
- Customization
- Assembly

DiverSE’s Seminar about SLE - May 28th, 2015
DiverSE’s Seminar about SLE: Program

- 10:00-10:30: Melange: a typing theory for language development (*Thomas Degueule*)
- 10:30-11:00: coffee break
- 11:00-11:30: Melange: language variability management (*David Mendez*)
- 11:30-12:00: Domain-Specific Metamodelling and language family (*José Galindo*)
- 12:00-13:00: lunch
- 13:00-14:15: Modular operational semantics for fundamental programming constructs (*Peter D. Mosses*)
- 14:15-14:30: The GEMOC Initiative (*Benoît Combemale*)
- 14:30-15:00: GEMOC: Reifying concurrency into language semantics (*Benoît Combemale*)
- 15:00-15:30: GEMOC: Trace management and model debugging (*Erwan Bousse*)
- 15:30-16:00: Adaptable Software Languages (*Marcelino Rodríguez*)
- 16:00-16:30: coffee break
- 16:30-17:00: Forces and Frictions in Metamodelling (*Guillaume Becan*)
- 17:00-17:30: Metamorphic DSLs (*Mathieu Acher*)
- 17:30-18:00: DSL for custom memory profilers (*Inti Gonzalez*)
This is YOUR seminar!

- Ask questions
- Provide feedback
- Make it interactive

- Only one constraint: respect the timing! 😊