Composition and concurrent execution of heterogeneous domain-specific models

A work part of the GEMOC initiative

Benoit Combemale
Associate Professor, University of Rennes 1
Research Scientist, INRIA
benoit.combemale@irisa.fr

Journée IDM et modèles scientifiques, 18 octobre 2013
TRISKELL - team

Research in software engineering.
- 9 faculty members
- ~40 researchers and engineers on projects
Application domains

Software intensive systems
Context

- Open environments
- Heterogeneous environments
- Distributed systems
- Dynamic systems
- Software intensive systems
Objectives

Model-driven software engineering to handle diversity
Approach

Continuous design for Heterogeneous, variable, adaptive within reasonable correctness boundaries
"DSLs are far more prevalent than anticipated"
[Hutchinson et al., ICSE’11]

AOMDE = Pleonasm!
A model is an abstraction of an aspect of reality for a given purpose

Change one aspect and Automatically re-weave:
From Software Product Lines...
..to Dynamically Adaptive Systems
Aspect Oriented Model Driven Engineering

Distribution

Security

Act1: Separation of Concerns => Language Engineering

Model
Abstraction Gap

Assembler

C, Java

Problem Space

DSLs

Solution Space
Domain-Specific (Modeling) Language?

• "Language **specially** designed to perform a task in a certain domain"

• "A formal processable language targeting at a **specific viewpoint or aspect** of a software system. Its **semantics and notation** is designed in order to support working with that viewpoint as good as possible"

• "A computer language that's targeted to a particular kind of problem, **rather than a general purpose language** that's aimed at any kind of software problem.”"
A GPL provides notations that are used to describe a computation in a human-readable form that can be translated into a machine-readable representation.

A GPL is a formal notation that can be used to describe problem solutions in a precise manner.

A GPL is a notation that can be used to write programs.

A GPL is a notation for expressing computation.

A GPL is a standardized communication technique for expressing instructions to a computer. It is a set of syntactic and semantic rules used to define computer programs.
General PL vs Domain SL

The boundary isn’t as clear as it could be. Domain-specificity is not black-and-white, but instead gradual: a language is more or less domain specific.

<table>
<thead>
<tr>
<th></th>
<th>GPLs</th>
<th>DSLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>large and complex</td>
<td>smaller and well-defined</td>
</tr>
<tr>
<td>Language size</td>
<td>large</td>
<td>small</td>
</tr>
<tr>
<td>Turing completeness</td>
<td>always</td>
<td>often not</td>
</tr>
<tr>
<td>User-defined abstractions</td>
<td>sophisticated</td>
<td>limited</td>
</tr>
<tr>
<td>Execution</td>
<td>via intermediate GPL</td>
<td>native</td>
</tr>
<tr>
<td>Lifespan</td>
<td>years to decades</td>
<td>months to years (driven by context)</td>
</tr>
<tr>
<td>Designed by</td>
<td>guru or committee</td>
<td>a few engineers and domain experts</td>
</tr>
<tr>
<td>User community</td>
<td>large, anonymous and widespread</td>
<td>small, accessible and local</td>
</tr>
<tr>
<td>Evolution</td>
<td>slow, often standardized</td>
<td>fast-paced</td>
</tr>
<tr>
<td>Deprecation/incompatible changes</td>
<td>almost impossible</td>
<td>feasible</td>
</tr>
</tbody>
</table>
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"
Domain-Specific (Modeling) Language

Editors (textuals, graphicals, …)

Documentation generators

Test generators

Simulators

Analyzers

Refactoring

Checkers (static & dynamics)

Translators

Compilers

Code generators

Etc.
External DSLs vs. Internal DSLs

• An **external** DSL is a completely separate language and with its own custom syntax/tooling support (e.g., editor, compiler)

  Language workbenches: Eclipse modeling (EMF, xText, GMF, Sirius, Kermeta, xCore/xTend…), DSL Tools, Meta Edit+, MPS, GME, Neverlang, Delite, etc.

• An **internal** DSL is more or less a set of APIs written on top of a host language

  Extension mechanisms: xTend’s active annotation, Scala’s LMS, extensions methods (e.g., xTend, Kotlin, Scala-Virtualized), plain-old java annotation or even fluent interfaces!

⇒ DSL technology is here (no excuse)
⇒ Related fields: generative programming, product lines
External DSLs vs. Internal DSLs

• Both internal and external DSLs have strengths and weaknesses
  • learning curve,
  • cost of building,
  • programmer familiarity,
  • communication with domain experts,
  • mixing in the host language,
  • strong expressiveness boundary
Textual DSLs vs. Graphical DSLs
Eclipse Modeling

EMF-based Tools
(68 matches in September, 2012)
The Kermeta Language Workbench [SoSyM’13]

• **Modular design of DSMLs**
  - One meta-language per language concern *(require)*
    - Ecore, OCL, Kermeta Action Language
    - But also: QVTo, fUML, Alf, Ket, Xsd…
  - Static introduction mechanism *(aspect)*

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

• *Current investigations: precise metamodeling, modeling in the large, language family, multi-platforms, model comprehension…*
Aspect Oriented Model Driven Engineering

Act2: Composition of Concerns => Global Engineering
Complex Software-Intensive Systems

- deal with multiple concerns
  → require *global analysis and execution*

- integrate heterogeneous parts
  → require *global service*

- manage evolution of concerns and the emergence of new concerns
  → require evolution and creation of tools and methods for software development
Heterogeneous Domain-Specific Modeling Languages
(AO) MDE  =  (AO) Modeling  +  Composition

Why? various intents
When? design vs. run time
What? homogeneous vs. Heterogeneous
Where? static vs dynamic introduction
How? symmetric vs. Asymmetric

=> different techniques for composition
Challenges

• *Model (Driven) Engineering*
  ➔ *(Software) Language Engineering*
  ➔ *Global (Software) Engineering*

• Language relationships should be capitalized
  ➔ from transformation to composition

• Global model coordination and analysis
  ➔ from design to runtime
On the Globalization of Modeling Languages
An Initiative...

Focuses on SLE tools and methods for interoperable, collaborative, and composable modeling languages

"On the Globalization of Modeling Languages" [GEMOC]
... Constantly Growing
The GEMOC Initiative is born!

An open initiative to

- coordinate (between members)
- disseminate (on behalf the members)

worldwide R&D efforts on the globalization of modeling languages

http://gemoc.org

- “Supporting coordinated use of DSMLs leads to what we call the globalization of modeling languages, that is, the use of multiple modeling languages to support coordinated development of diverse aspects of a system.”

- Advisory Board: Benoit Combemale (Fr.), Robert B. France (USA), Jeff Gray (USA), Jean-Marc Jézéquel (Fr.)
- Funded by complementary and successive projects (IP left to PCA of each projects)
The GEMOC Initiative: Objectives

Globalized DSMLs aim to support the following critical aspects of developing complex systems:

- **communication** across teams working on different aspects,
- **coordination** of work across the teams,
- and **control** of the teams to ensure product quality.

Current investigated application domains:

- Complex software-intensive systems: safety-critical embedded systems, cyber-physical systems, system of systems, dynamic adaptable systems
- Enterprise Architecture
The GEMOC Studio

• A language workbench for domain experts
  • DSMLs implementation and coordination

• A modeling workbench for domain designers
  • Heterogeneous modeling and simulation
ANR INS GEMOC (2012-2016)

"A Language Workbench for Heterogeneous Modeling and Analysis of Complex Software-Intensive Systems »

Tools and methods for the definition and coordination of heterogeneous executable modeling languages over heterogeneous models of computation

http://gemoc.org/ins
Concurrent execution of heterogeneous domain-specific models

• Metamodeling: Effective environments for the design and implementation of executable domain specific languages (e.g., Kermeta at Inria)
  • BUT these environments do not allow the integration of heterogeneous models of computation (concurrency, communication…)

• Models of computation: Effective environments to deal with the execution and analysis of models based on heterogeneous models of computation (e.g., Ptolemy at UC Berkeley, ModHel’X at Supélec)
  • BUT these environments do not allow adaptation to specific business/application domains
Bridging the gap between language theory and concurrency theory

fUML Implementation

Bridging the Chasm between Executable Metamodeling and Models of Computation (Benoit Combemale, Cécile Hardebolle, Christophe Jacquet, Frédéric Boulanger, Benoit Baudry), In 5th Int’l Conference on Software Language Engineering (SLE), 2012.
Concurrent execution of *homogeneous* domain-specific models

Timed Finite State Machine (TFSM) Implementation

... but also Logo, Actor model

Companion webpage: [http://gemoc.org/sle13](http://gemoc.org/sle13)

*Reifying Concurrency for Executable Metamodeling* (Benoit Combemale, Julien Deantoni, Matias V. Larsen, Frédéric Mallet, Olivier Barais, Benoît Baudry, Robert B. France), In 6th Int’l Conference on Software Language Engineering (SLE), 2013.
Concurrent execution of *heterogeneous* domain-specific models

**fUML and TFSM Coordination**

The GEMOC Studio: Current Status

• A workbench based on EMF
  • A language workbench for DSML Engineers
    • design and implement executable domain specific modeling languages with explicit model of computation, and (structural and behavioral) language interfaces
    • define structural and behavioral relations between DSMLs
  • A modeling workbench System/Software Engineers
    • model the heterogeneous aspects of complex software-intensive systems using various DSML
    • simulate and animate the coordinated execution of the heterogeneous models

• Current integrated technologies:
  • EMF, incl. Ecore, Ecore tools, and OCL
  • Kermeta, xTend, CCSL, Cometa and ECL
  • xText, Obeo Designer (Sirius)
  • GEMOC Execution Engine, Timesquare, Obeo Animator
Roadmap (ANR GEMOC)

• Language workbench
  • Explicit DSML interface
  • Meta-language for DSMLs coordination
  • Methodology for designing DSMLs and their coordination

• Heterogeneous model execution
  • Execution trace
  • Graphical animator design
  • Combining continuous time and discrete time
Conclusion and Perspectives

• Agile (Software Development) vs. (Agile Software) Development

• Model Driven Engineering
  ⇒ Software Language Engineering
  ⇒ Globalization of Modeling Languages

• The GEMOC Initiative