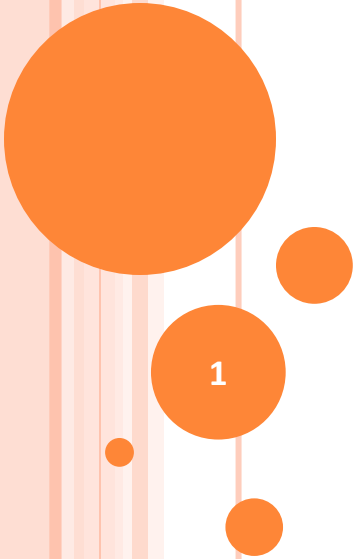

Modeling and Simulation Based Design

Domain-specific languages with JetBrains MPS - Kevin Buyl

JetBrains MPS

- MPS = Meta Programming System
- Implements the Language Oriented Programming (LOP) paradigm
- Created Traffic and TrafficLight languages
- Comparison with AToM³

Reference: Sergey Dmitriev, Language Oriented Programming: The Next Programming Paradigm (November 2004)

What is Mainstream Programming ?

- = using a general-purpose language (Java or C++) to build the application
- Steps:
  1. Think: conceptual model in your head
  2. Choose: choose a general-purpose language
  3. Program: write the solution by performing a difficult mapping from the conceptual model to the programming language ← bottleneck step

Reference: Sergey Dmitriev, Language Oriented Programming: The Next Programming Paradigm (November 2004)

What is Mainstream Programming ?

- Advantages: 
  - Implement every solution to a problem
- Disadvantages: 
  - Some solutions will take ages due to the nature of a general-purpose language (= unproductive)
  - Forces the programmer to think like a computer rather than have the computer think more like the programmer
  - Long gap between the idea of a solution and the solution itself (due to object-oriented design)
  - High-level idea is converted to low-level features of the language → the big picture is lost → reconstructing requires effort and time

Reference: Sergey Dmitriev, Language Oriented Programming: The Next Programming Paradigm (November 2004)

What is Language Oriented Programming ?

- = using a domain-specific language (DSL) for the problem
- Steps:
  1. Think: conceptual model in your head
  2. Choose: choose some specialized DSLs to write the solution
  3. Create: no appropriate DSLs for your problem → create one yourself
  4. Program: write the solution by performing a straightforward mapping from the conceptual model to DSL
  5. Compilation/generation of code (automated)

Reference: Sergey Dmitriev, Language Oriented Programming: The Next Programming Paradigm (November 2004)

What is Language Oriented Programming ?

- In other words:
  - Develop high-level, domain-oriented language
  - The development process then splits into two independent stages
    - Implement the system using this 'middle level' language
    - Implement a compiler, translator or interpreter for the language, using existing technology

Reference: M. P. Ward, Language Oriented Programming (October 1994)

What is Language Oriented Programming ?

- Advantages: 
  - Separation of concerns between design issues (domain-specific language) and implementation issues
  - High development productivity: problem-specific very high level language → a few lines of code are sufficient
  - Improves the maintainability of the design
  - Porting to a new operating system or programming language becomes simplified
  - Opportunity for reuse (reuse of the middle level languages)

Reference: M. P. Ward, Language Oriented Programming (October 1994)

What is Language Oriented Programming ?

- Disadvantages: 
  - The strength of DSLs, domain specificity, is also their weakness
  - What we want is different languages for every specific part of the program that can work together → need to create, reuse, modify and extend/mix languages → can be done in JetBrains MPS

Reference: Sergey Dmitriev, Language Oriented Programming: The Next Programming Paradigm (November 2004)

What is a language in LOP ?

- 3 main components:
  - Structure = abstract syntax (what concepts are defined and how are they arranged)
  - Editor = concrete syntax (how it should be presented)
  - Semantics (how should it be interpreted and how should it be transformed into executable code)

Reference: Sergey Dmitriev, Language Oriented Programming: The Next Programming Paradigm (November 2004)

JetBrains MPS

- MPS doesn't use plain text form
  - Normal programs: compile a program → text parsed into a abstract syntax tree (AST)
  - Major drawback: loss of extensibility
  - The language (language grammar) cannot be extended by programmers
  - New features can make the language ambiguous
  - In MPS: the program and all language concepts are directly stored in a structured graph (everything is a node, even language constructs itself)
  - Due to this feature it is possible to extend/mix languages

Reference: Sergey Dmitriev, Language Oriented Programming: The Next Programming Paradigm (November 2004)

JetBrains MPS

- Project = organizational unit
- Projects consist of 1 or more modules, which themselves consist of models
- Several types of modules: solutions, languages, generators, …
- A language consists of several models, each defining a certain aspect of the language: structure, editor, actions, constraints, …
- A language can extend another language
- Models consist of root nodes (represent top level declarations) and non-root nodes
- The basic notions of MPS:
  - Nodes, concepts and languages
- A concept defines the "type“ of node
  - Specifies children, properties and references
  - Concept declarations form an inheritance hierarchy

Reference: http://confluence.jetbrains.net/display/MPSD1/MPS+User%27s+Guide

JetBrains MPS

- Features of the language (concepts) defined in structure aspect → abstract syntax
- The editor aspect defines the layout of cells for each concept in the language → concrete syntax
- The generator of a language defines a transformation to other languages (e.g. Java) → semantics

Reference: Sergey Dmitriev, Language Oriented Programming: The Next Programming Paradigm (November 2004)

JetBrains MPS

- Example:
  - Traffic and TrafficLight languages
  - Concept(s) of TrafficLight language used in Traffic language → extending/mixing languages
  - Generator of Traffic language transforms traffic network to an executable Java class → operational semantics
  - Demo …

Comparison with AToM³

	AToM³	JetBrains MPS
Representation	Visual	Textual/Visual
Abstract syntax	Classes in Class Diagram model Associations	Concepts in structure aspect (children, references)
Concrete syntax	Icons/images for class instances	Editor aspect (cell lay-out)
Code generation	Button in formalism	Generator language
Simulation	Button in formalism Rewrite rules	Only after generation (in Java)
Constraints	Multiplicities Constraints in code	Multiplicities Constraint aspect
Multiple formalisms	Yes	Yes
Extending languages **Weaving languages**	No	Yes
Change in meta-model → change in model	No	Names (after refactoring)
User-friendliness (+/++/+++)	+	+++

Conclusion & Future work

- Implementing, extending and mixing languages in JetBrains MPS is quite easy
- Some languages (e.g. Traffic) are better designed in a graphical tool like AToM³ for better simulation
- Future (possible extra features) :
  - Queuing for road segments
  - More realistic execution (rather than a stepwise execution)
  - Generator to transform the traffic network in the high-level language to a Java Swing application→ the same or better results then the AToM³ implementation