Defining Domain Specific Languages with UML Profiles

Dr. Darius Silingas
Principal Trainer, MagicDraw UML
darius.silingas@nomagic.com
http://www.magicdraw.com

• Each domain has its own concepts..
• Are the domain specific modeling languages any new?
Domain Specific Languages

- System Modeling Language – **SysML**
- Business Process Modeling Notation – **BPMN**
- Department of Defense Architectural Framework – **DoDAF**
- Ministry of Defense Architectural Framework – **MoDAF**
- NATO Architecture Framework – **NAF**
- **Zachman Framework** for EA and IS Architecture
- DSLs for telecom, finance, and other industries
- Company-specific DSLs

Modeling Language for All Domains

- General purpose software modeling language
  - Standardized for 10 years already
  - Very popular but still not used by everyone
  - UML extension mechanisms are not any new!
UML ≠ Universal Modeling Language

What’s the Change all of a Sudden?

• UML tools are still there
  – However, DSL tools built on UML have arrived!

• What has changed?
  – Requirements are fully brought into the modeling game
  – Now all types of engineers can stop using 200+ different tools (most just for charting or being overly complex) and notations and rely on DSL on top of UML
DSL Implementation by OMG

• OMG offers two ways of DSL implementation:
  – **Meta Object Facility** (MOF) language as a tool for creating new modeling languages
  – **UML with profiles**. Due to better support in modeling tools, the latter approach is dominant in practice.

OMG 4-level Metamodel Architecture

M3

<table>
<thead>
<tr>
<th>MOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>corresponds to</td>
</tr>
</tbody>
</table>

M2

<table>
<thead>
<tr>
<th>UML metamodel</th>
</tr>
</thead>
<tbody>
<tr>
<td>based on</td>
</tr>
</tbody>
</table>

M1

<table>
<thead>
<tr>
<th>User model (UML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>is modeled using</td>
</tr>
</tbody>
</table>

M0

<table>
<thead>
<tr>
<th>Real world</th>
</tr>
</thead>
<tbody>
<tr>
<td>corresponds to</td>
</tr>
</tbody>
</table>
Profile and Stereotype

• Profile is a specific package, which contains a list of stereotypes
• Each stereotype (M1 level) extends a set of meta-classes (M2 level)
  – Modeler can apply stereotypes on the instances of these meta-classes while modeling at M1 level, i.e. drawing UML diagrams
• Stereotypes extend semantics of standard UML elements and may also provide tag definitions, custom appearance icons, and OCL constraints

Creating Profiles

Perform the following sequence of actions for creating a new profile in MagicDraw UML:

1. Create new profile data element
2. Add stereotypes to profile
   • Use class diagram for modeling stereotypes
3. Export profile as module
Modeling stereotypes in MagicDraw

- **Stereotype** extends meta-classes
  - Stereotype defines:
    - Name
    - Icon [optional]
    - Tag definitions [optional]
  - They can be generalized from other stereotypes or have associations between them

Best-Practices for Modeling Profiles

- **Naming style**
  - Upper/Lower case, consistent names, joining words
  - UML specifies only 1 naming space for stereotypes!
- **Icons**
  - Create consistent set of intuitive icons
- **Granularity**
  - Keep different domains in separate profiles
  - You can use multiple profiles in one project
- **Avoid using small proprietary profiles !!!**
Advantages of UML Profiles

- Reuse existing UML tool infrastructure for DSL
  - Repository
  - Drawing engines
  - Persistence
  - Analysis
  - Integrations
- Standard exchange format
- References between business and software models

Disadvantages of Regular UML Profiles

- No way to shrink, just to extend
- Notation change possibilities are limited
- Clumsy usage of tagged values
Why DSL is “Specific”

• **Domain-Specific**
  – notation
  – properties
  – ownership/nesting rules
  – types restrictions
  – relations

How to make UML-Based DSL Efficient?

• **The main goal:**
  – Hide UML elements everywhere
  – Use DSL elements as new meta-types

• **The tools for achieving that:**
  – Properly designed profiles
  – DSL validation rules
  – Scalable icons for shapes and paths
  – Custom diagrams
  – Customization of specification dialogs
  – Customization of model browser
  – Custom rules for relationships
  – Custom rules for ownership
Workflow for Creating DSL

1. Identifying concepts in the domain;
2. Creating UML profile with stereotypes and tag definitions;
3. Creating DSL validation rules;
4. Customizing DSL modeling environment;
5. Implementing DSL-specific code generator.

– Don’t forget to prepare and use samples for testing if your creating profiles really supports modeling practical issues!

1. Identifying Concepts in the Domain

• Software development company runs a number of projects. Employees participate in the projects as project leaders or simple project members. Projects are decomposed into tasks and project members work on those tasks. Each employee is managed (supervised) by a project leader. Projects and tasks require coordination that can be done by project leader.
2. Creating UML Profile

1. Create a new project
2. Create a new profile *Organization Profile* in data browser
3. Create a new class diagram for modeling stereotypes
4. Define stereotypes
   - Name them according to identified domain concepts
   - Map organization concepts to UML meta-classes
5. Model associations between stereotypes
6. Specify stereotype properties (tag definitions)
7. Assign icons to stereotypes
8. Define custom paths for stereotyped relationships
9. Share the profile
3. Create DSL Validation Rules

- You may create domain-specific validation rules suite
  - Based on OCL constrains
  - Can be used for evaluating model
- For example:
  - Project must have at least one managing project leader
  - Employee cannot be involved in more than 2 projects
  - One project leader can manage at most 7 people in the project
  - …

Creating Domain-Specific Constraints

- It is often necessary to create OCL constraints for domain-specific languages implemented as UML profiles
- In order to restrict the modeling rules inherited from UML 2 meta-model, you may assign OCL constraints to stereotypes and use model validation against the created user models.

```ocl
class component : Element { owner.oclIsKindOf(product) or owner.oclIsKindOf(component) }
class product : Package { self.ownedElement->exists{x|oclIsKindOf(component))}
```
Specifying OCL Constraint Details

Invalid Domain-Specific Model
4. Customizing DSL Modeling Environment

- Custom Diagram Wizard
  - allows creating your own diagram types for custom profile, with your own toolbars, stereotyped elements, symbol styles and custom smart manipulators. Such customization is saved in the special “diagram descriptor”, that could be exchanged between users, allowing them use your own custom diagrams.

- Domain Specific Language (DSL) Customization Engine
  - allows “tuning” domain specific profiles, customizing multiple GUI, model initialization, and semantic rules, creating your own Specification dialogs. DSL customization is model-driven approach, based on UML profiling. Customization is saved as a UML model.

- Advanced UML Profiling
  - allows using some profiling enhancements that are not defined in UML, but that help to solve some common problems like tag grouping, unwanted stereotypes and tags hiding etc.

Customize Paths for Stereotyped Relationships

- Stereotyped relationship notation is customized by specifying its icon, which allows to select from predefined line, begin and end styles and specify custom begin and end icons.
Define DSL Customizations

- Create a new package *Customizations* inside profile
- Create a new class diagram for modeling customizations
- Define customizations
  - Create classes with corresponding to stereotype names
  - Apply stereotype `<<Customization>>` for the classes
  - Specify customization properties

Specifying Customization Properties

- Specify customization rules either through specification properties or directly through Tags for *Customization* stereotype
  - Application of bootstrapping technique: Customization stereotype is customized itself!
What You Can Specify in DSL Customization

<table>
<thead>
<tr>
<th>More Usual</th>
<th>Less Usual</th>
</tr>
</thead>
<tbody>
<tr>
<td>• More Usual</td>
<td>• Less Usual</td>
</tr>
<tr>
<td>– Generic Representation</td>
<td>– Custom model initialization</td>
</tr>
<tr>
<td>• Meta-class representation text</td>
<td>• Default super classes and interfaces</td>
</tr>
<tr>
<td>• Keyword representation text</td>
<td>– Connection rules</td>
</tr>
<tr>
<td>• Context actions for property changing</td>
<td>• Allowed meta-classes or stereotypes for relationship ends</td>
</tr>
<tr>
<td>– Custom Specification dialogs</td>
<td>• Stereotypes that must be applied on relationship ends</td>
</tr>
<tr>
<td>• Grouping into nodes</td>
<td></td>
</tr>
<tr>
<td>• Native UML properties on/off</td>
<td></td>
</tr>
<tr>
<td>• Always visible custom properties</td>
<td></td>
</tr>
</tbody>
</table>

Company Customization

```xml
<Customization>
   <Company>
       customizationTarget = Company
       suggestedOwnedDiagrams = "Organization Chart"
       inShortcutMenu = isBig
       multiLineTextProperties = address
       hiddenOwnedDiagrams = "Any Diagram"
       suggestedOwnedTypes = Project
       suggestedOwnedTypes = Employee
       hideMetatype = true
       possibleOwners = Package
       hiddenOwnedTypes = Element
       quickApplyingFor = Class
       disallowedRelationships = Generalization
   </Company>
</Customization>
```

```xml
<propertyGroup>
   Projects{properties = "projects", useAsNode, columns = "name", "leader"}
</propertyGroup>

<propertyGroup>
   Employees{properties = "employees", useAsNode, columns = "name", "salary"}
</propertyGroup>
```
Create Custom Diagram

- Run Diagrams | Customize Diagrams | Create, and then follow these steps:
  1. Specify diagram type and icon
  2. Specify modules
  3. Specify toolbars
  4. Specify toolbar buttons
  5. Specify symbol properties
  6. Specify smart manipulators

The New Custom Organization Chart Diagram
Sample Organization Chart Diagram

No Magic, Inc.

- Applying DoDAF Training Course
- MagicDraw 12.5 Development

- Programming
- Requirements

- Daniel B.
  - experience = "20+ years"
  - salary = 10000
  - company = No Magic, Inc.

- Mindaugas R.
- Nerijus J.

- No Magic, Inc.
  - employees = Daniel B., Mindaugas R., Nerijus J.
  - isBig = true
  - address = "7304 Alma Drive, Suite 600, Plano, TX 75025"
  - projects = Applying DoDAF Training Course, MagicDraw 12.5 Development

Company Specification Dialog

- No Magic, Inc.
  - Projects: Applying DoDAF Training Course, MagicDraw 12.5 Development
  - Employees: Daniel B., Mindaugas R., Nerijus J.
  - Address: 7304 Alma Drive, Suite 600, Plano, TX 75025
  - Projects: Applying DoDAF Training Course, MagicDraw 12.5 Development
  - Employees: Daniel B., Mindaugas R., Nerijus J.
5. Implementing DSL-Specific Code Generator

Code Generation Cartridge

UML model → Java, C++, XML Schema, DDL, HTML

Thank You for Attention!
Keep In Touch!

darius.silingas@nomagic.com
http://www.magicdraw.com