EVALUATION OF THE DIAGRAM DEFINITION STANDARD FOR VISUAL LANGUAGE SPECIFICATION

DRT/LIST/DILS/Laboratoire d'Ingénierie dirigée par les modèles pour les Systèmes Embarqués

June 22nd 2015
Visual modeling language specification

- **Semantic**
  - Specified with models
  - Usable by computers

- **Abstract syntax**
  - Rely on "by example" interpretation
  - Not usable by computers

- **Visual aspect**
  - Concrete syntax
  - Graphical vocabulary

Specified with models and usable by computers.
GMF-Tooling provides a formalism to specify a modeling language, and then generate the appropriate diagram editor.

- Domain model (*.ecore)
- Graphical definition (*.gmfgraph)
- Tools definition (*.gmftool)
- Mapping model (*.gmfmap)
- Generator model (*.gmfgen)
- Diagram editor (plugin Eclipse)

Rely on "by example" interpretation
Not usable by computers
GMF-Tooling provides a formalism to specify a modeling language, and then generate the appropriate diagram editor. This process has never been used because the GMFGen model had to be adapted anyway.
Diagram Definition support the MVC pattern required to provide tooling for a visual language [OMG12].
UMLDI exemplify the use of DD for the specification of the UML concrete syntax [OMG12].
OBJECTIVES OF THE EVALUATION APPROACH

Is the DD standard sufficient to fully specify the concrete syntax of a language?

Can we generate fully featured spec-compliant tooling?
Restrictions:

- 1-1 mapping to the abstract syntax
- Reuse UML abstract syntax from Papyrus
- Reuse UML rendering implementation from Papyrus
What we need to know to build the GMFGen model:
- The different kind of graphical element’s behavior
- The different kind of graphical relations
- Connections with the abstract syntax (ModelFacet)
- Connections with graphical vocabulary (FigureViewMap)
Does the DI meta-model answer the GMFGEN requirements?

- Only three kinds of graphical elements
- Use of a UML profile
Do we know enough about graphical relations?

Extract of UMLDI meta-model

Not enough information on graphical relations

- Person
  - Attributes
  - Operations
  - NestedClassifiers
  - Enumeration Litterals
  - Enumeration Litterals
Do we know enough about graphical relations?

Extract of UMLDI meta-model

Not enough information on graphical relations

Syntax enrichment
Do we know enough about abstract syntax connections?

Each graphical element can be linked to any abstract syntax elements.

Extract of DI meta-model

From that, can we automatically deduce connections between graphical inclusion relations and abstract syntax containments?
Does DD answer the GMFGEN requirements?

From that, can we automatically deduce connections between graphical relations and abstract syntax containments?
From that, can we automatically deduce connections between graphical relations and abstract syntax containments?
Does DD answer the GMFGEN requirements?

From that, can we automatically deduce connections between graphical relations and abstract syntax containments?

Diagram:

Abstract syntax:

- State
  - entry
  - doActivity
  - exit

- Behavior

Development of an algorithm to analyze the whole UML meta-model.

→ 172 containments cases including 34 ambiguous.
→ Considering the UML redefinition mechanism allows to solve only one case.
→ Language designer or user intervention is required.
UMLDI enriched
• Graphical element distinction
UMLDI enriched
- Graphical element distinction
- Syntax enrichment
**UMLDI enriched**

- Graphical element distinction
- Syntax enrichment
- Connections with abstract syntax elements
UMLDI enriched
- Graphical element distinction
- Syntax enrichment
- Connections with abstract syntax elements
- Graphical element’s behavior distinction
Tool generation process

- Execution of the prototype on UMLDI
- Execution of the GMFT M2T transformation
- M2M Transformation
- Diagram editor within Eclipse
- Eclipse plugin
- Deployment

Containments computation

CEA | 22 JUNE 2015 | PAGE 21
Generated diagram editor

Only the simple cases are handled

Graphical elements are correctly handled

Graphical relation are correctly handled
• Improve our understanding of the DD standard: what we can do, what are the limitations.

• Proposal of a formalism to fully specify the graphical syntax of a language using the DD standard

• Development of a prototype to validate the proposal.
Only the very simple cases are handled.

The graphical vocabulary is not exploited.

Ad-hoc graphical element’s behavior identification.
• Handle the graphical vocabulary using a modular QVTo DI\rightarrow\text{DG} transformation.

• Handle n-m mapping to the abstract syntax using a framework for incremental transformation.

• Handle real label specification.

• Provide a more general and flexible mechanism for graphical element’s behavior identification, or

• Provide a declarative framework to model UI interactions.
1. Get UMLDI element
2. Produce DG element
3. Render DG element
4. Repaint related FigureViewMaps

instanceOf
Thank you for your attention

Any questions?
<table>
<thead>
<tr>
<th>Kind of element handled (UMLDI meta-model)</th>
<th>Java class</th>
<th>Kind of element generated (GMFGen model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI::Diagram</td>
<td>DIGenDiagram</td>
<td>GenMultiFacetedNode</td>
</tr>
<tr>
<td>DI::Shape</td>
<td>DiGenShape</td>
<td>GenMultiFacetedNode</td>
</tr>
<tr>
<td>DI::Shape « affixed »</td>
<td>DiGenAfxNode</td>
<td>GenAffixedNode</td>
</tr>
<tr>
<td>DI::Shape « compartment »</td>
<td>DiGenCompartment</td>
<td>GenCompartment</td>
</tr>
<tr>
<td>DI::Shape « label »</td>
<td>DIGenLabelNode</td>
<td>GenLabelNode</td>
</tr>
<tr>
<td>DI::Shape « internallabel »</td>
<td>DIGenLabel</td>
<td>GenLabel</td>
</tr>
<tr>
<td>DI::Edge</td>
<td>DIGenEdge</td>
<td>GenLink</td>
</tr>
</tbody>
</table>