Goal-Oriented Measurement: Comprehensibility of Model Transformations

UKSMA/COSMIC International Conference on Software Metrics & Estimating October 27th/28th 2011

Shekoufeh Kolahdouz Rahimi, Kevin Lano

Shekoufeh.kolahdouzrahimi@kcl.ac.uk, kevin.lano@kcl.ac.uk
Model Driven Engineering

- Model-Driven Engineering (MDE) is a software development methodology which focuses on creating and exploiting domain models.
- The MDE approach is meant to increase productivity by maximizing compatibility between systems, simplifying the process of design and promoting communication between individuals and teams working on the system.
- MDE has been promoted as a solution to handle the complexity of software development by raising the abstraction level.
- More attention is paid to the quality aspects in Model-Driven Engineering along with growing importance of modelling in software development.
The concepts of Model-Driven Architecture and Model-Driven Development use model transformations as a central element, principally to transform high-level models such as Platform Independent Models (PIMs) to Platform-Specific Models (PSMs).

Model transformations are employed for variety of reason within model driven development: to improve model quality, to systematically apply design patterns for refactoring, to map models from one language to another.

A large number of model transformation languages and tools have been developed across the research community. However, there are no guidelines on how to select appropriate notations for particular model transformation tasks, and no comprehensive comparisons of the relative merits of particular approaches.
Different styles for Model Transformations

- **Declarative**: transformations are described abstractly, as a mathematical relations between source and target models.

- **Imperative**: transformations are defined as programs which explicitly define the details of how a source model is transformed into a target model.

- **Hybrid**: style is a combination of declarative and imperative, as a wide-spectrum specification language in which a declarative description can be refined.
Transformation Paradigms
UML-RSDS (Reactive System Development Support)

- UML-RSDS is a UML-based specification language, consisting of UML class diagrams, state machines, activities, sequence diagrams, and a subset of OCL.

- Used as the specification language of an automated Model-Driven Development approach, Constraint-Driven Development (CDD), by which executable systems can be synthesised from high-level specifications.
Measurement in Model Transformation

- Young research discipline and most effort is invested in applications and in improving Model Transformation techniques and Tools

- Analysis technique need to be developed to assist in the maintenance process

- Proper Tool support is required for further adaptation of MDE by the industry
Comprehensibility in Model Transformation

- The ability in which specification and implementation of model transformation can be understood and analysed.

- A comprehensive specification is crucial for an efficient maintenance process.

- The more comprehensive the specification, the less effort will be required to understand the transformation properties.
Case Studies

- Tree To graph Transformation
- UML to relational database schemas
- Class diagram refactoring
Tree to Graph Transformation

Tree
- name: String
  - {frozen, identity}
  - parent

Node
- name: String
  - {frozen, identity}

Transformation

Edge

Source
- *

Target
- *
UML to relational database schemas
Class diagram refactoring

Transformation
Goal/Question/Metric paradigm

- Developed in response to the need for a goal-oriented approach that would support the measurement of processes and products in the software engineering domain
- Supports a top-down approach to defining the goals behind measuring software processes and products
- The GQM Paradigm supports a bottom-up approach to interpreting data based on the previously defined goals and questions
- The GQM Paradigm may be seen as purely an approach for choosing metrics
Measurement of Comprehensibility in Model Transformation

- Size
- Complexity
- Modularity
Comprehensibility

What is the overall size of transformation
- Total number of lines of code

How complex is the transformation
- Total number of calls
- Total number of recursion calls
- Maximum depth of recursive loop
- Total number of non-recursion calls
- Maximum depth of non-recursive loop

How modular is the transformation
- Sum of interconnectivity between distinct modules (Coupling)
- Some of interconnectivity inside each individual module (Cohesion)

What is the overall size of transformation
- How complex is the transformation
- How modular is the transformation

The diagram illustrates the factors that determine the overall size, complexity, and modularity of a transformation.
Size Metrics of Case studies

<table>
<thead>
<tr>
<th>Approach</th>
<th>LOC In First Case Study</th>
<th>LOC In Second Case Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>QVT</td>
<td>27</td>
<td>180</td>
</tr>
<tr>
<td>VIATRA</td>
<td>22</td>
<td>180</td>
</tr>
<tr>
<td>Kermeta</td>
<td>15</td>
<td>174</td>
</tr>
<tr>
<td>ATL</td>
<td>12</td>
<td>86</td>
</tr>
<tr>
<td>UML-RSDS</td>
<td>11</td>
<td>144</td>
</tr>
</tbody>
</table>
Complexity

- Complexity of a specification can be measured by analysing its call graph.

- A call graph is a directed graph that represents calling relationships between subroutines in a program.

- In the call graph each node represents functions or rules and edges represent calls between nodes.

- The size of the call graph affects the complexity of the program. The greater the number of arcs in the call graph, the higher is the dependency between different parts of the program, and so the greater is the complexity.
Comprehensibility

What is the overall size of transformation
  - Total number of lines of code

How complex is the transformation
  - Total number of calls
  - Total number of recursion calls
  - Maximum depth of recursive loop
  - Total number of non-recursion calls
  - Maximum depth of non-recursive loop

How modular is the transformation
  - Sum of interconnectivity between distinct modules (Coupling)
  - Some of interconnectivity inside each individual module (Cohesion)

What is the overall size of transformation
  - How complex is the transformation
  - How modular is the transformation

How complex is the transformation
  - Total number of calls
  - Total number of recursion calls
  - Maximum depth of recursive loop
  - Total number of non-recursion calls
  - Maximum depth of non-recursive loop

How modular is the transformation
  - Sum of interconnectivity between distinct modules (Coupling)
  - Some of interconnectivity inside each individual module (Cohesion)
## Complexity of UML to relational database transformations

<table>
<thead>
<tr>
<th>Approach</th>
<th>Total number of calls</th>
<th>Total recursive calls</th>
<th>Depth2</th>
<th>Total non-recursive calls</th>
<th>Depth1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kermeta</td>
<td>22</td>
<td>9</td>
<td>2</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>VIATRA</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>QVT</td>
<td>10</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>UML-RSDS</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>ATL</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>
Modularity

- Enhances the maintainability and reusability of transformation.

- Introduced principles to encapsulate sets of rules inside each module. We provide evaluation based on coupling and cohesion of each transformation specification.

- Low coupling results in high cohesion, readability and maintainability while lack of cohesion may be associated with low modularity and high complexity.
Comprehensibility

What is the overall size of transformation
- Total number of lines of code

How complex is the transformation
- Total number of calls
- Total number of recursion calls
- Maximum depth of recursive loop
- Total number of non-recursion calls
- Maximum depth of non-recursive loop

How modular is the transformation
- Sum of interconnectivity between distinct modules (Coupling)
- Some of interconnectivity inside each individual module (Cohesion)
Modularity Rules

We have applied three rules to restructure our examples, the aim is to maximise the cohesion and minimise the coupling to increase analysability and usability of the system.

- All rules with inheritance relation should be located inside the same module. Inheritance is a dedicated reuse mechanisms between rule which make strong dependency between them.

- Mutual calls, (rules with iteration relation) are encapsulated inside one module as they have a substantial influence on each other.

- The rules with hierarchical relationship are located inside one module.
Conclusion

- The process measurement framework helps by providing specific goals, questions and metrics.
- Metrics evaluate size and complexity of different model transformation approaches on different case studies.
- It would expected Kermeta to be larger in terms of size but QVT is larger as it has a number of syntax in declaration.
- We found out that QVT and Kermeta have large complexity in terms of the number of recursive calls which hindered the verification.
- Principles has been introduced to improve the modularity and reduce the complexity metrics.
Questions?