Model-Driven Requirements Engineering and Quality

João Araújo

(In collaboration with Miguel Goulão and Ana Moreira)

NOVALINCS, Universidade Nova de Lisboa, Portugal
Requirements Modeling Issues

- Quality of goal-oriented models
  - How complex and complete are goal models?
  
- Is the transformation effort worth it?
  - From Activity to Sequence Diagrams
Evaluating the quality of goal models: the case of KAOS
Introduction

- Goal-oriented Requirements Engineering (GORE)
  - Great impact and importance in the Requirements Engineering community
  - Provide expressive model elements for requirements elicitation and analysis
    - i*, KAOS, GRL
- But…
  - The models can quickly become very complex
  - Manage the accidental complexity of the models is a challenge
- We need to identify refactoring opportunities to improve the modularity of those models, and consequently reduce their complexity
Objectives

• To provide a tool supported metrics suite, targeted to the measurement and analysis of the
  ▫ Complexity of goal models, for identifying modularity refactoring opportunities
  ▫ Completeness of these models, facilitating the modeler’s work
• The identification of such opportunities can be useful during development, where a better structure can lead to a sounder system understanding
  ▫ If performed in a timely fashion, this is likely to contribute to relevant costs savings through the reduction of the model’s accidental complexity
  ▫ Refactoring opportunities identification is also an asset in the context of preventive maintenance, as a facilitator for future requirements changes
• Measuring the current status of a model, and its level of completeness at a given time, can help in calculating the estimated effort required to finish the modelling process
The Approach

- Improve the modularity of goal models and reduce their complexity and completeness

**Metrics Set**
- Informal definition
- Formal definition

**KAOS modelling tool**
- KAOS goal models creation
- Metrics collecting

**Metrics evaluation**
- Case studies set
- Statistics analysis
About the metrics suite

- The metrics suite is integrated in an eclipse-based KAOS editor, so that metrics can be computed during the requirements modelling process, whenever the requirements engineer requests them.
- The metrics are defined using the Object Constraint Language (OCL) upon the KAOS metamodel.
- This makes our metrics set easily extensible, as improving the metrics set can be done by adding new OCL metrics definitions.
Goal models for the elevator system
Obstacles

- Passengers informed of elevator direction
- Elevator direction updated few seconds before next stop
- Elevator direction unreadable
  - Inadequate LED size
  - Elevator direction unreadable by blind people
  - Insufficient display contrast
- Resolution Link
  - LEDs at least 2in high
  - Elevator direction announced by voice
KAOS and Model Quality

For a medium-size (gym) system with

- 5 main functionalities
- 15 agents
- 212 sub-goals

- Is this model complete?
- How complex is this model?
- Is this complexity really necessary?

- GORE aimed at large scale systems
- Models can become really hard to understand
We need to…

• Analyse the extent to which a model is close to being complete
• Assess model complexity to identify model refactoring opportunities, e.g.:
  ▫ Models may have a too deep goal hierarchy
  ▫ Agents may have too many responsibilities
• Prevent unanticipated extra costs in the development phase
  ▫ Better management of the completeness and complexity of the models
Tool support with metrics for KAOS Models

- Tool supported approach in the metrics-based evaluation of the completeness and complexity of KAOS goal models
- The developer can measure the current status of his model and take on corrective actions, during model construction.
- The tool support is based on the integration of a KAOS editor with a KAOS metrics suite and
  - targeted to the requirements elicitation process,
  - it can also support post-mortem analysis from which lessons can be learned for future projects.
- Metrics are formally defined using OCL
  - ModularKAOS developed in MDD on top of Eclipse
    - We validate the metrics set and their implementation by extending an existing tool for editing KAOS goal models
Approach outline

- Metrics **identification** using the Goal-Question-Metric approach
- Metrics **definition** using OCL
- Metrics **evaluation** with real-world case studies
  - Often used as example of best practices using KAOS
- KAOS models analysis with metrics support
# Goal: KAOS models completeness evaluation

<table>
<thead>
<tr>
<th>Question</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Q1.</strong> How close are we to completing the assignment of all goal responsibilities to agents?</td>
<td><strong>PLGWA.</strong> Percentage of Leaf Goals With an Agent.</td>
</tr>
<tr>
<td><strong>Q2.</strong> How detailed is the goal model with respect to objects?</td>
<td><strong>PLGWO.</strong> Percentage of Leaf Goals With an Object.</td>
</tr>
<tr>
<td><strong>Q3.</strong> How close are we to completing the resolution of all the goal obstacles?</td>
<td><strong>PLOWS.</strong> Percentage of Leaf Obstacles With a reSolution.</td>
</tr>
<tr>
<td><strong>Q4.</strong> How detailed is the goal model with respect to operations?</td>
<td><strong>PLGWOOp.</strong> Percentage of Leaf Goals With an Operation.</td>
</tr>
<tr>
<td><strong>Q5.</strong> How well supported are the operations in the goal model?</td>
<td><strong>POpWA.</strong> Percentage of Operations With an Agent.</td>
</tr>
</tbody>
</table>
Goal: KAOS models complexity evaluation

<table>
<thead>
<tr>
<th>Question</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Q6.</strong> Does an agent have too much responsibility in the model?</td>
<td><strong>ANLG.</strong> Number of Leaf Goals per Agent.</td>
</tr>
<tr>
<td><strong>Q7.</strong> Does a leaf goal have too many/few objects?</td>
<td><strong>GNO.</strong> Number of Objects per Goal.</td>
</tr>
<tr>
<td><strong>Q8.</strong> How difficult is it to understand a model, with respect to the number of refinement levels?</td>
<td><strong>MD.</strong> Model Depth.</td>
</tr>
<tr>
<td><strong>Q9.</strong> How complex is a model, with respect to its goal refinements?</td>
<td><strong>RNSG.</strong> Root Number of Sub-Goals.</td>
</tr>
</tbody>
</table>
modularKAOS: partial metamodel
Q1 - How close are we to completing the assignment of all goal responsibilities to agents?

<table>
<thead>
<tr>
<th>Name</th>
<th>PLGWA – Percentage of Leaf Goals With an Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informal definition</td>
<td>Percentage of leaf goals that have an associated agent in the model.</td>
</tr>
<tr>
<td>Formal definition</td>
<td>context KAOS</td>
</tr>
<tr>
<td></td>
<td>def: PLGWA(): Real = self.NLGWA() / self.NLG()</td>
</tr>
<tr>
<td>Pre-condition</td>
<td>context KAOS::PLGWA()</td>
</tr>
<tr>
<td></td>
<td>pre: self.NLG() &gt; 0</td>
</tr>
<tr>
<td>Comments</td>
<td>If there are no leaf goals the result is undefined. This requires:</td>
</tr>
<tr>
<td></td>
<td>NLG – Number of Leaf Goals</td>
</tr>
<tr>
<td></td>
<td>NLGWA – Number of Leaf Goals With an Agent</td>
</tr>
<tr>
<td>Recommendation</td>
<td>In a complete model, all leaf goals should be assigned to an agent.</td>
</tr>
</tbody>
</table>
Computing % of leaf goals with an agent

\[ PLGWA = \frac{NLGWA}{NLG} \]

\[ PLGWA = \frac{4}{5} = 0.8 \]
Evaluation of KAOS Models
Completeness
Percentage of Leaf Goals with an Agent

Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Completeness of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Percentage of Leaf Goals with an Object

Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Completeness of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Percentage of Leaf Obstacles with a reSolution

Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Completeness of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Percentage of Leaf Goals with an Operation

Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Completeness of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Percentage of Operations with an Agent

Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Complements of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Complexity
Number of Leaf Goals per Agent

Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Completeness of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Objects per Goal

Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Completeness of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Completeness of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Completeness of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Discussion (Completeness)

- Most models handle responsibility assignment of leaf goals to agents.
- Objects are not frequently used.
- When obstacles are specified, we find a big variation (from 0% to 100%) of the percentage of obstacles with a resolution.
- Operations are even more rarely used than objects.
- Only two of the case studies model the assignment of operations to agents, showing this is a fairly unexplored modeling feature.

Discussion (complexity)

- Relatively few leaf goals assigned to an agent
  - Do not attribute too many responsibilities to a single agent
- Assigning objects to goals is a mostly unexplored feature of models
- Model depth varies much less than the number of model elements, suggesting a fairly consistent state of practice with respect to what is considered an adequate model decomposition level
- Big variations in the case studies, concerning the number of subgoals defined in each model
  - The average number is around 40 subgoals, although in one of the examples it is over 200 goals.

Espada, Goulão, Araújo, “A Framework to Evaluate Complexity and Completeness of KAOS Goal Models”, CAiSE 2013, Valencia, Spain
Scenarios

• Scenarios are often modeled with activity models, in an early stage of development.
  ▫ Later, sequence diagrams should also be used to detail object interactions.
  ▫ But in practice time to market makes difficult using both models
  ▫ Also, the migration from activity diagrams to sequence diagrams is a repetitive and error-prone task.

• MDE can help streamlining this process, through transformation rules.
  ▫ But, since the information in the activity model is insufficient to generate the corresponding complete sequence model, manual refinements are required
Refinement effort

• Let’s compare the relative effort of building the sequence diagrams manually with that of building them semi-automatically.
MIGRATING FROM ACTIVITY TO SEQUENCE MODELS

• Rule 1: Generating Objects in Sequence Models. Boundary and control objects are created by default in sequence models with the name of the activity model that represents the scenario under study.
Transformation rules

- **Rule 2: Generating Messages in Sequence Models.** Each *activity* in an activity model is mapped into a message
  
  ▫ **Rule 2.1: Object flows.** The direction of the flow connecting an activity to an object indicates if it is a *read* or a *write* operation
  
  ▫ **Rule 2.3: Swimlanes.** When a message is generated from an activity that is inside an actor’s swimlane, the source object of that message is of type actor.
Transformation rules

- Rule 3: Generating Sequence Model Operators.
  - Rule 3.1: Generating **PAR Operators**. A PAR operator is created in the sequence model for each pair of *fork-join* elements is in the activity model.
  - Rule 3.2: Generating **ALT, OPT and LOOP Operators**. Algorithms for graphs with cycle detection mechanisms can be used to detect cycles in an activity model.
Refining Sequence Models

- Sequence models are more fine-grained than activity models and, hence, additional information should be provided to the generated model.
- Typical refinements:
  - add arguments and types
  - decompose a message to a set of messages
  - add return messages
  - add variables
  - initialize guards
  - delete undesired elements
Tool support

- We implemented a plug-in for the Eclipse platform to support the transformations described in the previous section.
- We used the Eclipse Modeling Framework (EMF) and UML2 plug-in for Eclipse.
APPLICATION TO THE MOBILE MEDIA CASE STUDY
Activity model: Send Media via SMS

- **Actor** "User"
  - Select Send Media via SMS
  - Select Media
  - Select target number
  - Answer retry

- **System** "Mobile Media"
  - Show Message "Select Media to Send"
  - Get Media
  - Show Message "Select a destination Number"
  - Send message
  - Show Message "Try Again?"
  - Create message
  - Show Message "Message Sent Successfully"
  - Show Message "Error sending message"

- **Variables**
  - mediaObject: Media
  - messageObject: Message

- **Decision Points**
  - retry
  - [yes] SendOK
  - [no]

- **Flow Control**
  - [yes]
  - [no]
Transformation to SD

1. Select sendMediaViaSMS
2. Select sendMediaViaSMS
3. Show message ("Select Media to Send")
4. Get Media
5. Send message
6. Loop [sendOK = no & retry = yes]
   - Show message ("Error Sending Message")
   - Confirmation()
Refinement
Validation and discussion

• Our research hypothesis is that the usage of our approach allows a significant effort reduction, when compared to building sequence models from scratch.

• Consider the following actions made in a sequence model:
  • removal of any kind of element;
  • insertion of a variable/argument name;
  • insertion of a variable/argument type;
  • insertion of an operator (PAR, ALT, etc.) and respective guard conditions;
  • insertion of an object and its name;
  • insertion of a message and the corresponding procedure call name (if necessary).
Refinement effort

- Assume all these actions have a similar time cost and assign one unit of time cost to each of them.
- If we were to build the refined model from scratch, this would require 72 editions.
- In contrast, if we start by generating the non-refined model and then apply a sequence of editions, we only need 32 editions (30 additions + 2 removals) to obtain the refined model.
- The effort has decreased from 72 to 32 editions, which corresponds to a reduction of around 55%.
- As part of our validation effort, more scenarios were developed and sequence models generated successfully, in the context of the Mobile Media case study.
- To simplify, we consider all types of action as having the same cost.
Insertions and deletions

- The effort to create the listed sequence models from scratch is 270 editions (the number of elements), while the total cost to refine them is 97 (the number of refinements).
- The effort has decreased from 270 to 97 editions, a value that shows a significant improvement of around 64%. We can also observe that most refinement actions are insertions (88) rather than deletions (9).
- This means that most of the automatically generated elements are correct.
- Additional edits to the generated models are dominated by insertions, with relatively few deletions.
  - In fact, 5 of the scenarios required no deletions at all.
<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Number of Elements</th>
<th>Number of Insertions</th>
<th>Number of Removals</th>
<th>Number of Refinements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send Media via SMS</td>
<td>72</td>
<td>30</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>Send Media via Internet</td>
<td>74</td>
<td>31</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>Create Media</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Delete Media</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Create Album</td>
<td>20</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Add Media to Album</td>
<td>32</td>
<td>6</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>List Media</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Configure Favourite Media</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>List Albums</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Play Media</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Delete Album</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>270</td>
<td>88</td>
<td>9</td>
<td>97</td>
</tr>
</tbody>
</table>
Number of Elements vs Number of Refinements
Discussion

- The number of edits required for building a sequence model from the activity model decreased by around 64%
  - (i) a reduction in the effort building the sequence model,
  - (ii) increased traceability among models (through the semi-automatic translation rules),
  - (iii) error prevention when migrating from different scenarios notations, and
  - (iv) the relative number of refinement actions does not increase unexpectedly as the number of scenario model elements increases.
This is an example, of course there are several threats to validity…

• We considered an interaction kind of system, results can be different with other kinds of system.

• The usage of a non-weighted effort unit for insertions and deletions which is agnostic to whether these are made in the context of a model built from scratch, or in the context of a model refinement is also a threat.

  ▫ This simplification should not affect significantly the outcome of this validation, as deletions represent less than 10% of all the refinements.

• When editing a generated model we might also want to consider the **time invested in understanding the existing model**, before making changes to it.

  ▫ We need to assess the extent to which that effort is significantly different from the one spent when editing a model built manually.
THANX!!

QUESTIONS?