FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

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

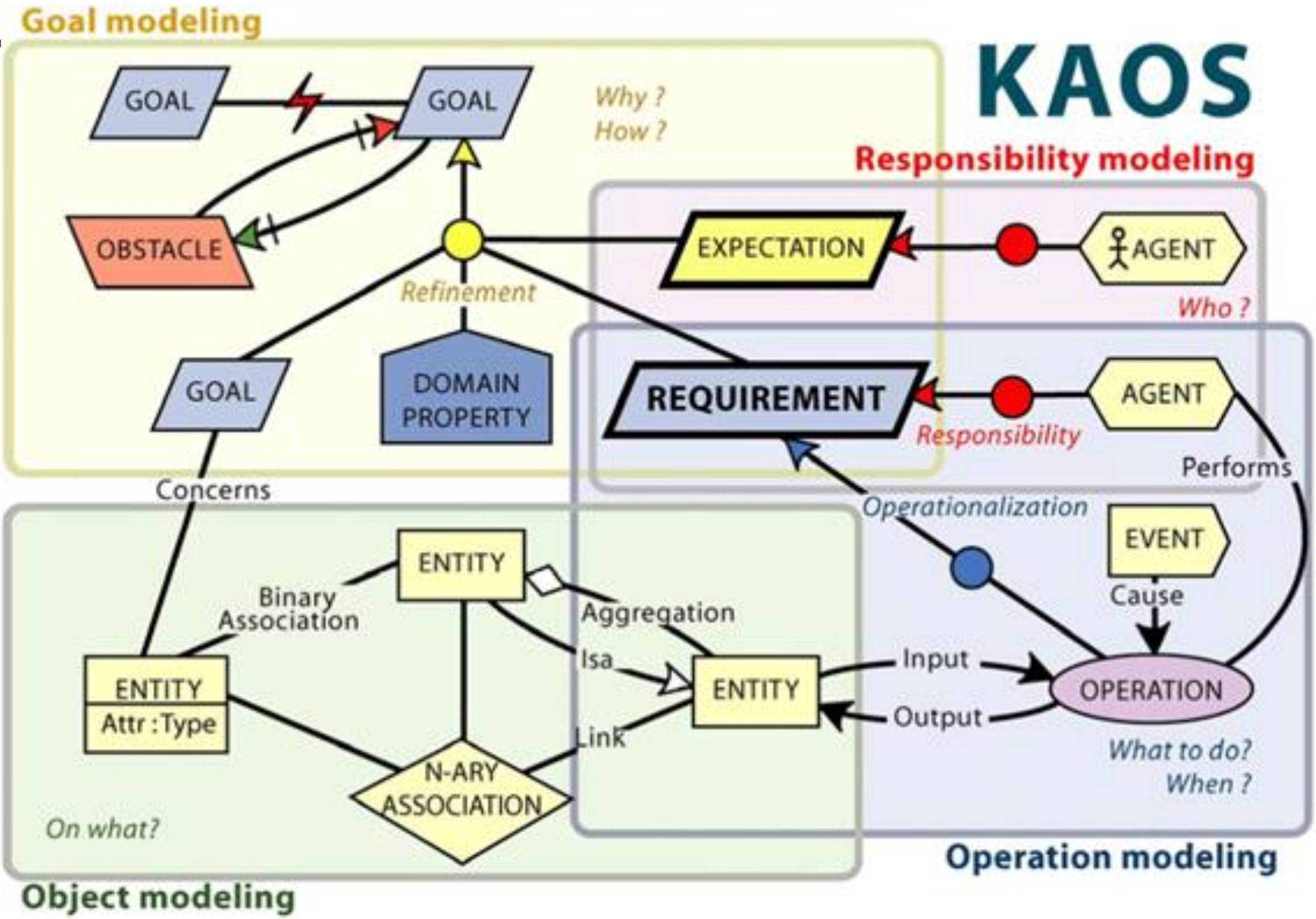
The Approach

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

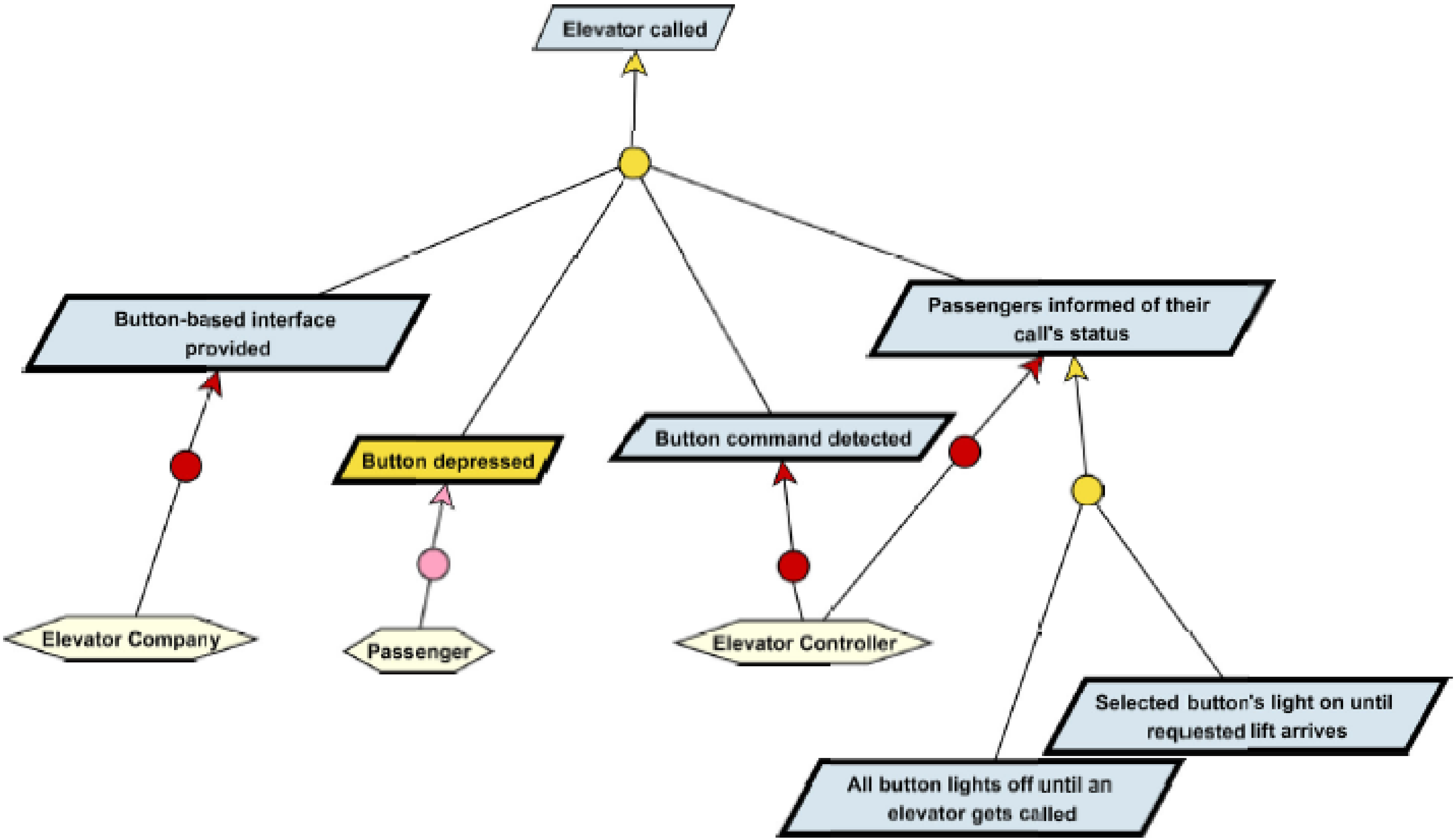
About the metrics suite

- 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

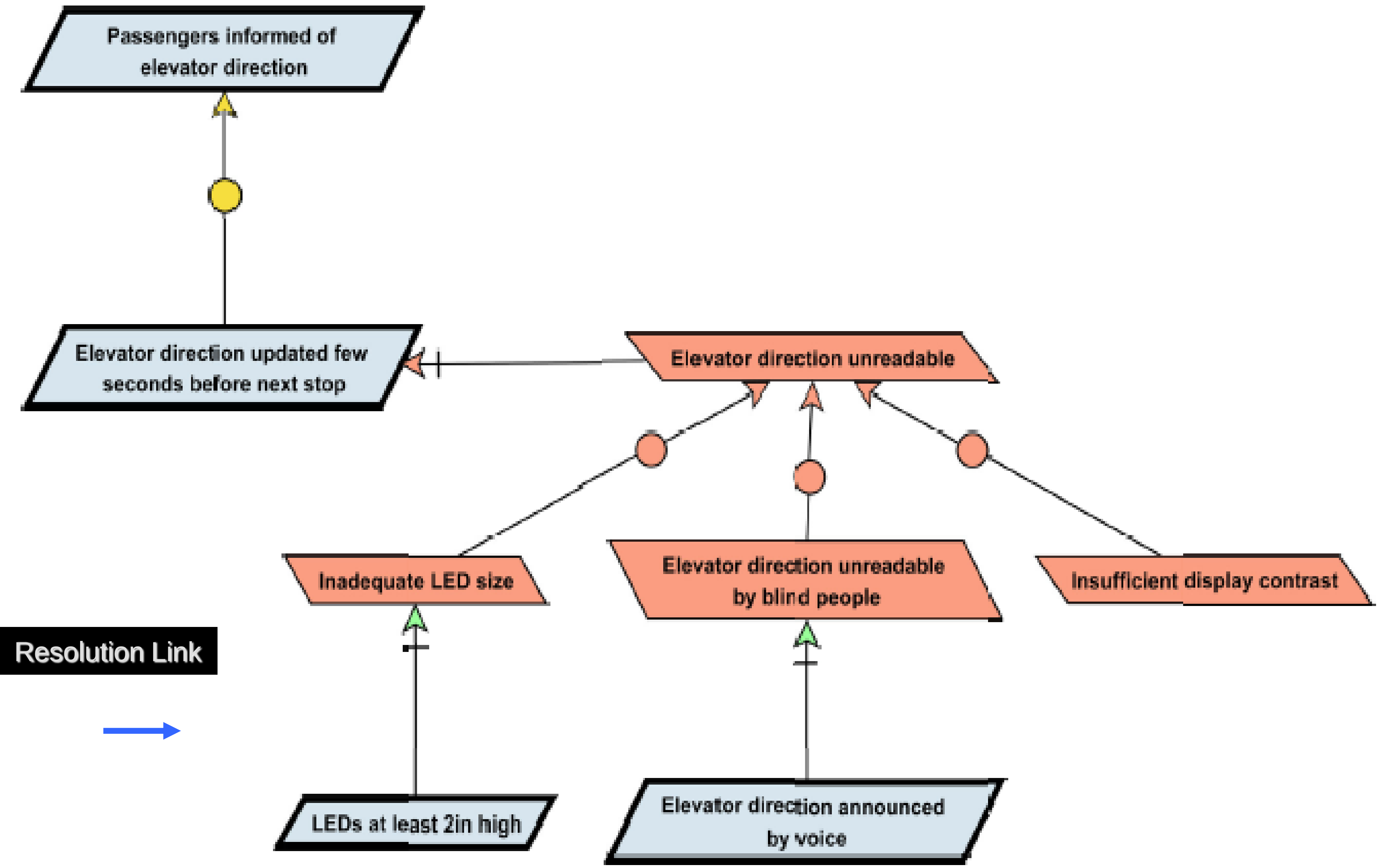


Goal modeling
KAOS
GOAL
GOAL
Why ?
How ?
Responsibility modeling
OBSTACLE
EXPECTATION
AGENT
Refinement
Who ?
GOAL
DOMAIN PROPERTY
REQUIREMENT
AGENT
Responsibility
Performs
Concerns
Operationalization
EVENT
ENTITY
Binary Association
Aggregation
Cause
Isa
Input
ENTITY
Attr : Type
ENTITY
OPERATION
Output
Link
What to do?
When ?
N-ARY ASSOCIATION
On what?
Operation modeling
Object modeling

Goal models for the elevator system



Obstacles

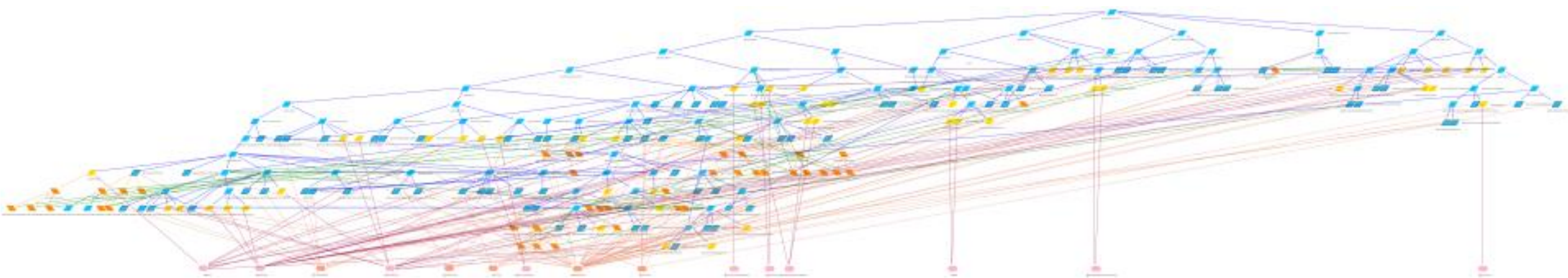

Passengers informed of elevator direction
Elevator direction updated few seconds before next stop
Elevator direction unreadable
Inadequate LED size
Elevator direction unreadable by blind people
Insufficient display contrast
Resolution Link
LEDs at least 2in high
Elevator direction announced by voice

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

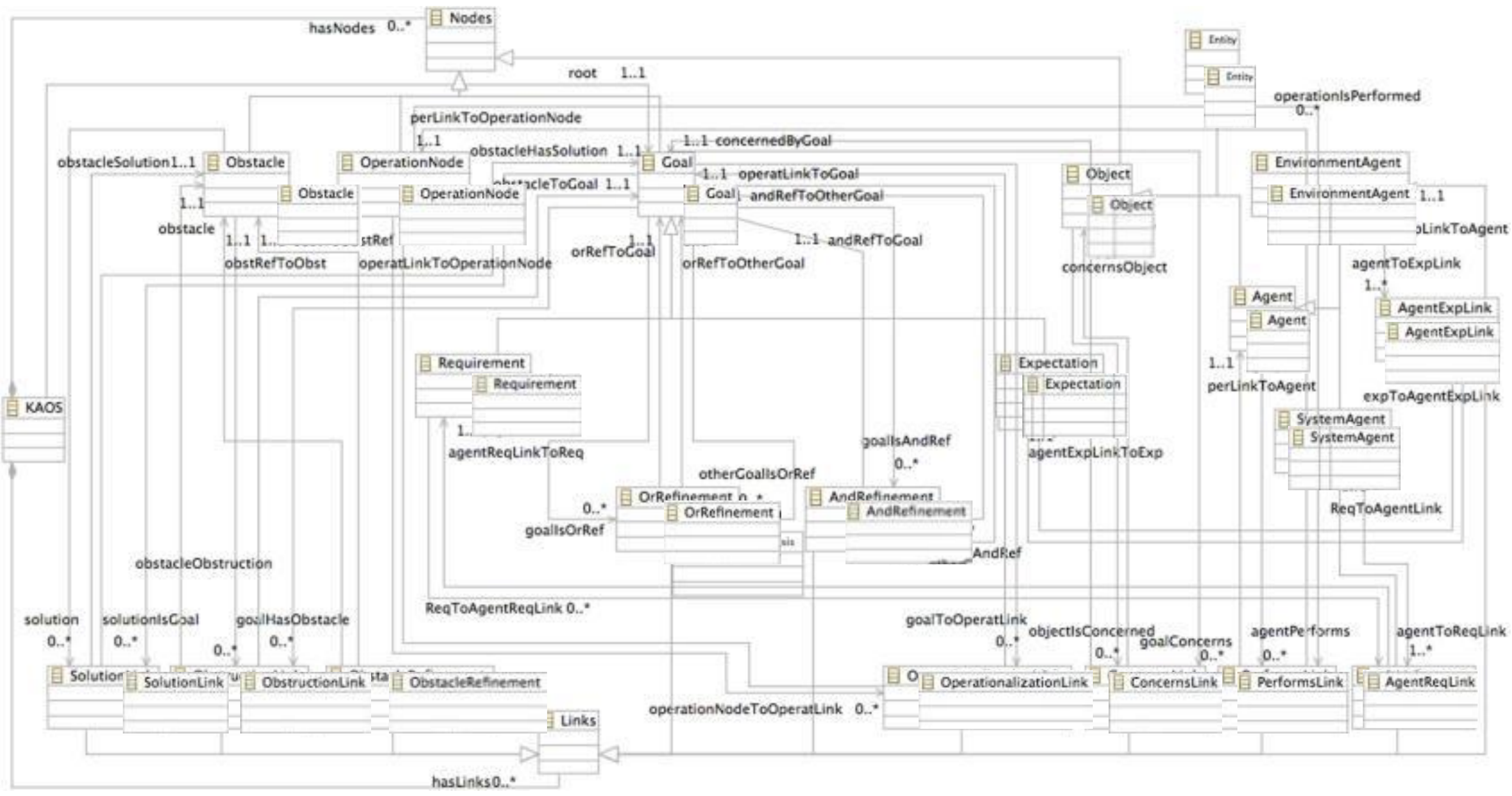
Goal: KAOS models completeness evaluation

Question	Metric
Q1. How close are we to completing the assignment of all goal responsibilities to agents?	**PLGWA**. **P**ercentage of **L**eaf **G**oals **W**ith an **A**gent.
Q2. How detailed is the goal model with respect to objects?	**PLGWO**. **P**ercentage of **L**eaf **G**oals **W**ith an **O**bject.
Q3. How close are we to complete the resolution of all the goal obstacles?	**PLOWS**. **P**ercentage of **L**eaf **O**bstacles **W**ith a re**S**olution.
Q4. How detailed is the goal model with respect to operations?	**PLGWOp**. **P**ercentage of **L**eaf **G**oals **W**ith an **Op**eration.
Q5. How well supported are the operations in the goal model?	**POpWA**. **P**ercentage of **Op**erations **W**ith an **A**gent.

Goal: KAOS models complexity evaluation

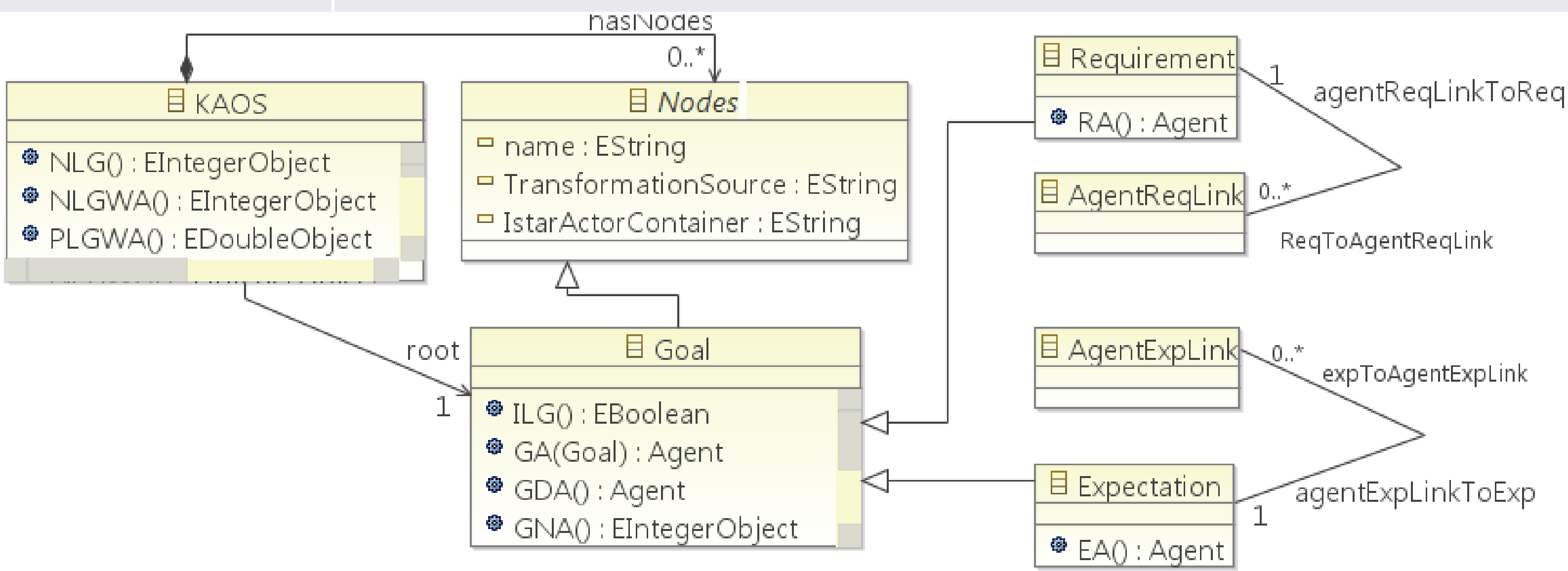
Question	Metric
Q6. Does an agent have too much responsibility in the model?	**ANLG**. **N**umber of **L**eaf **G**oals per **A**gent.
Q7. Does a leaf goal have too many/few objects?	**GNO**. **N**umber of **O**bjects per **G**oal.
Q8. How difficult is it to understand a model, with respect to the number of refinement levels?	**MD**. **M**odel **D**epth.
Q9. How complex is a model, with respect to its goal refinements?	**RNSG**. **R**oot **N**umber of **S**ub-**G**oals.

modularKAOS: partial metamodel



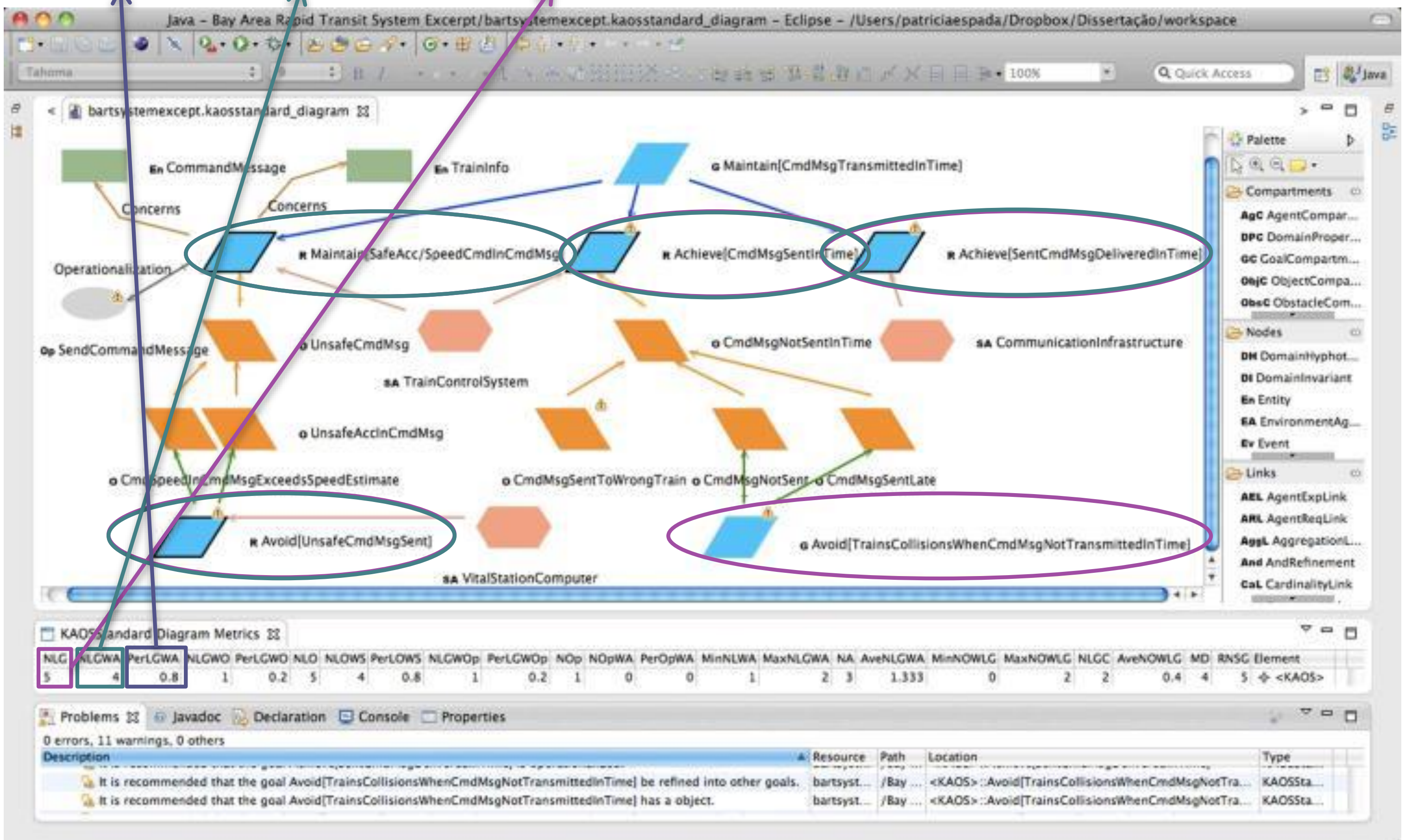
Metrics definition

Q1 - How close are we to completing the assignment of all goal responsibilities to agents?	
Name	**PLGWA – P**ercentage of **L**eaf **G**oals **W**ith an **A**gent
Informal definition	Percentage of leaf goals that have an associated agent in the model.
Formal definition	**context KAOS** **def: PLGWA(): Real** = self.NLGWA() / self.NLG()
Pre-condition	context KAOS::PLGWA() pre: self.NLG() > 0
Comments	If there are no leaf goals the result is undefined. This requires: **NLG – N**umber of **L**eaf **G**oals **NLGWA**– **N**umber of **L**eaf **G**oals **W**ith an **A**gent
Recommendation	In a complete model, all leaf goals should be assigned to an agent.



Computing % of leaf goals with an agent

PLGWA = NLGWA / NLG **PLGWA = 4 / 5 = 0.8**



Evaluation of KAOS Models



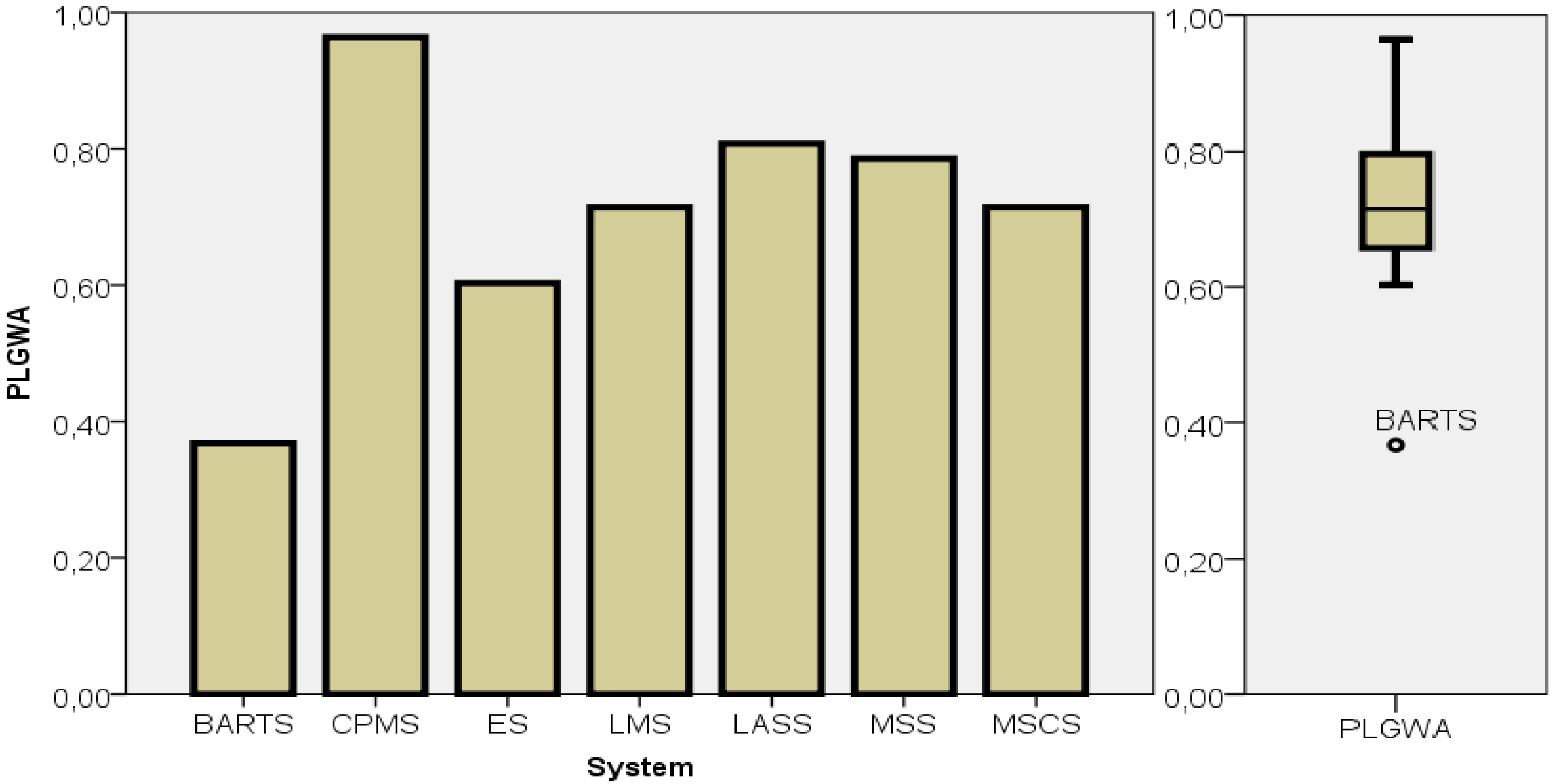

BARTS MSCS ES CPMS



LMS LAS MS

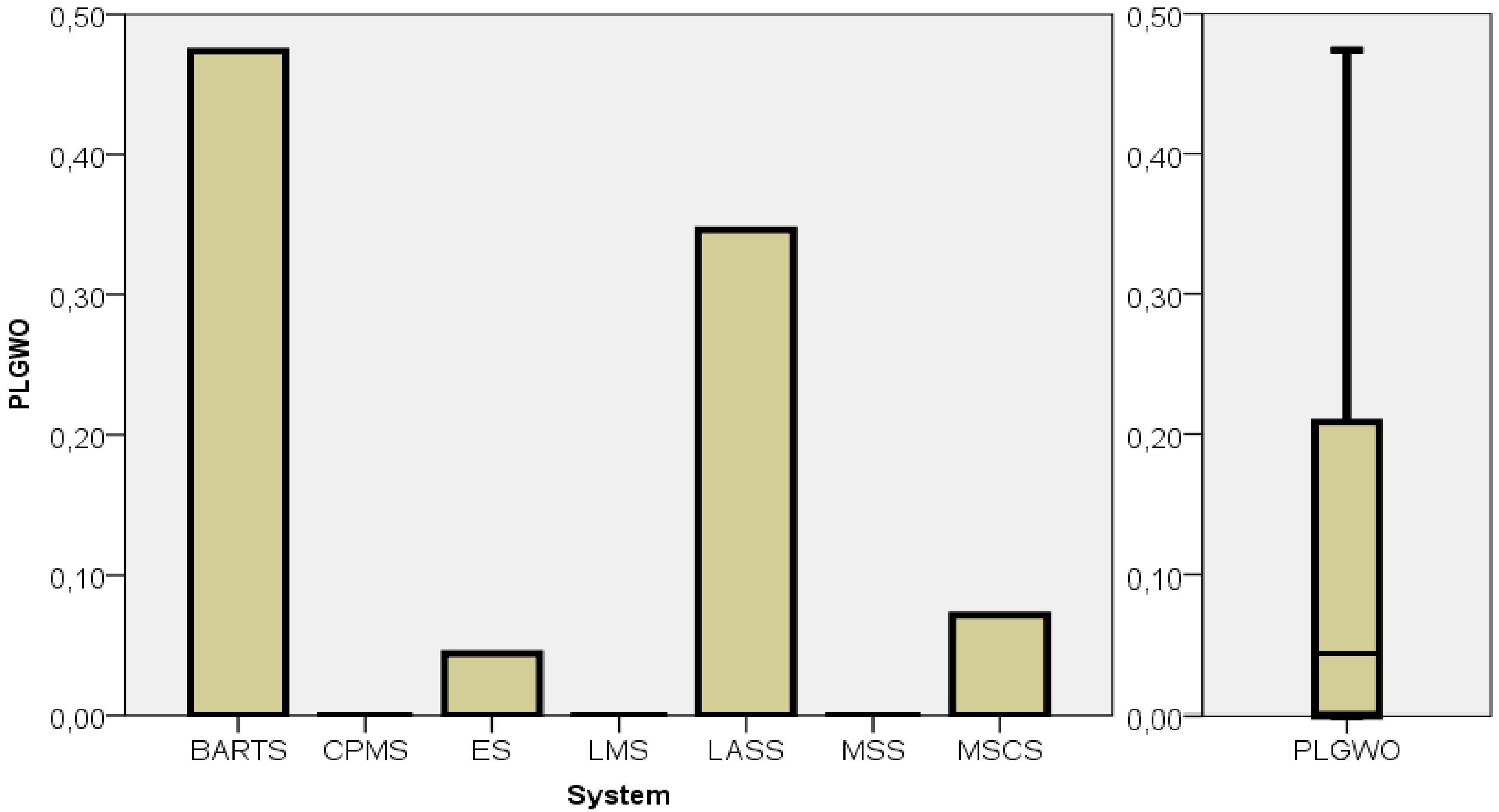
Completeness

Percentage of **L**eaf **G**oals **w**ith an **A**gent



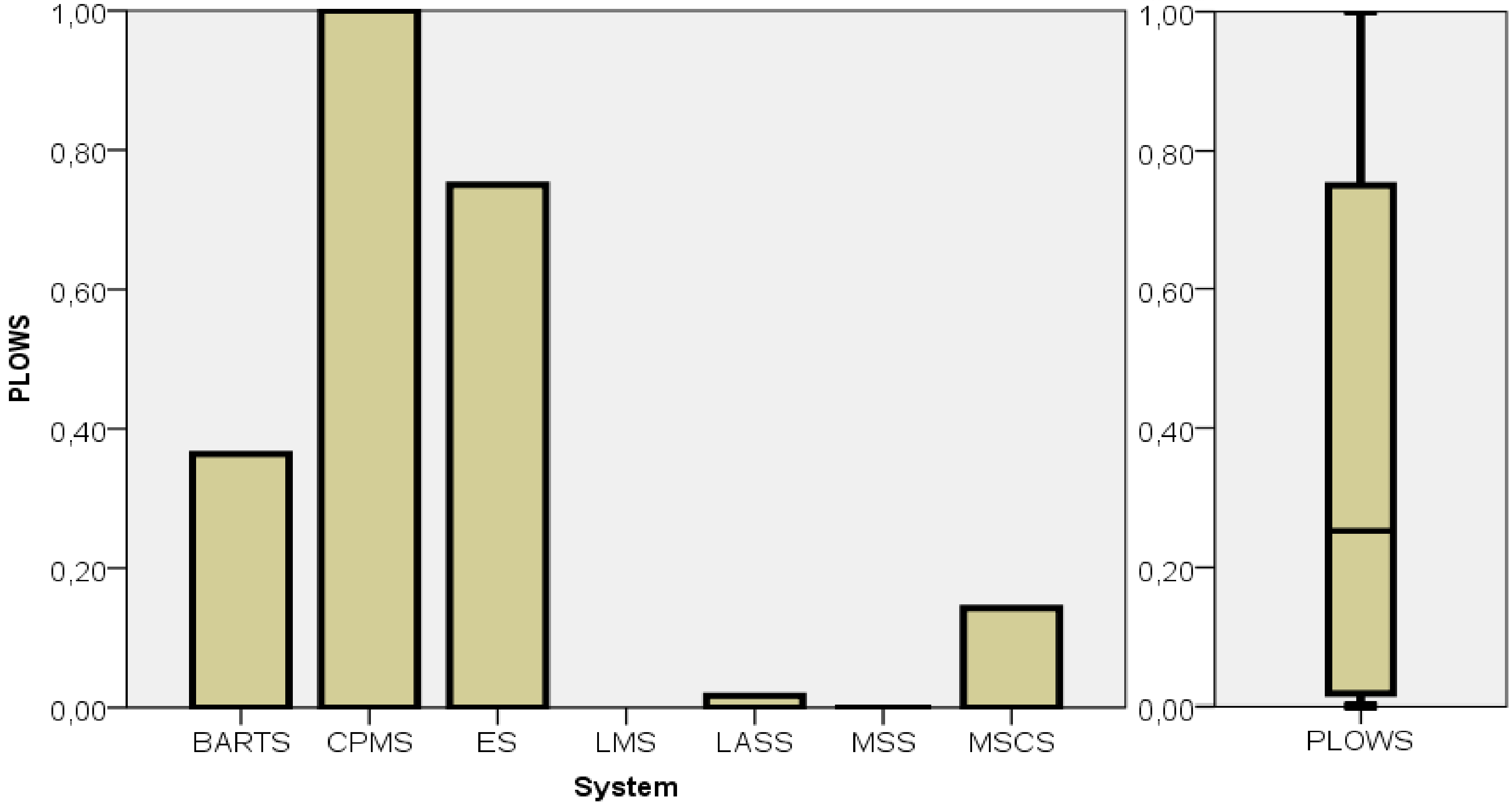
Espada, Goulão, Araújo, "*A Framework to Evaluate Complexity and Completeness of KAOS Goal Models*", CAiSE 2013, Valencia, Spain

Percentage of Leaf Goals with an Object



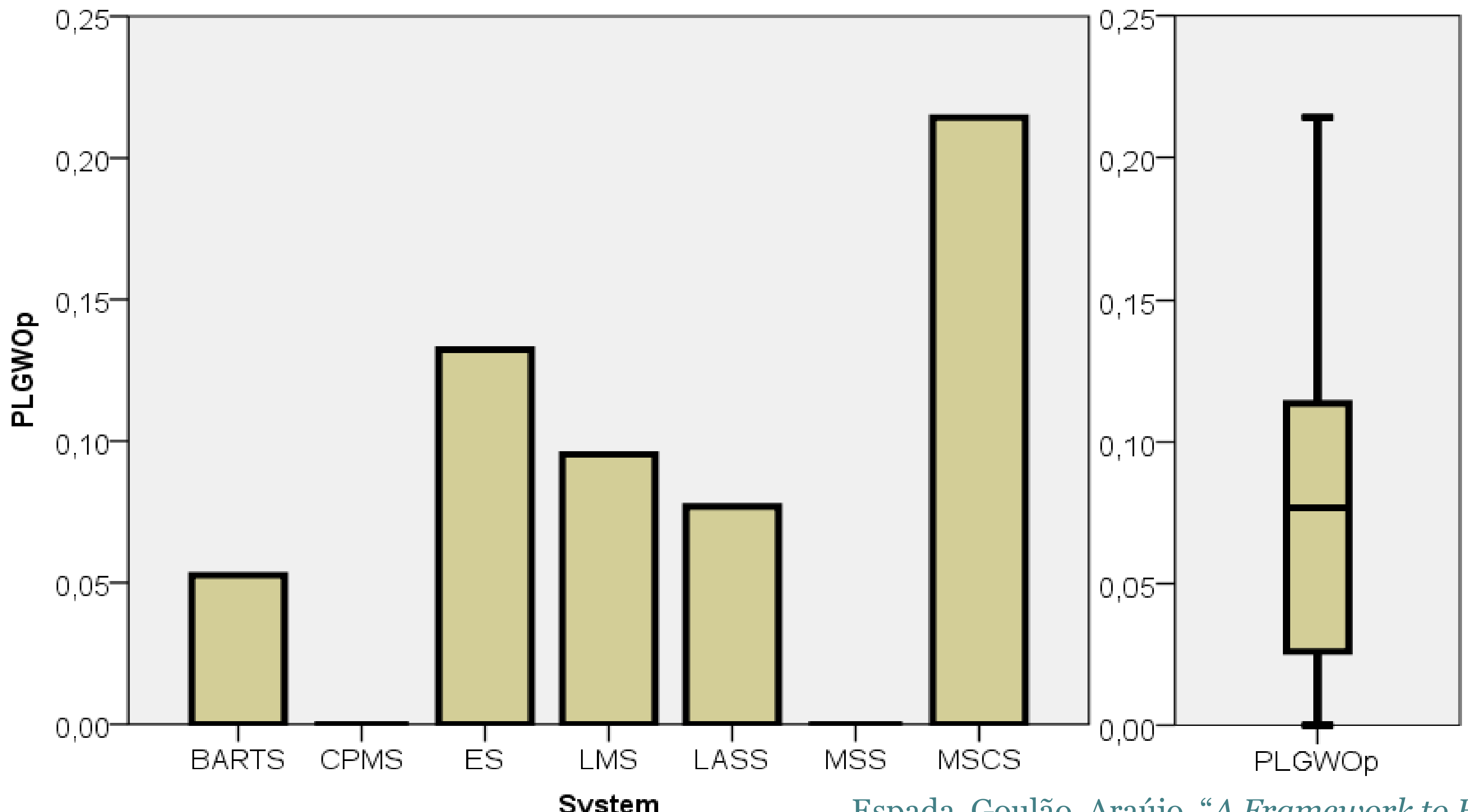
Espada, Goulão, Araújo, "*A Framework to Evaluate Complexity and Completeness of KAOS Goal Models*", CAiSE 2013, Valencia, Spain

Percentage of **L**eaf **O**bstacles **w**ith a re**S**olution



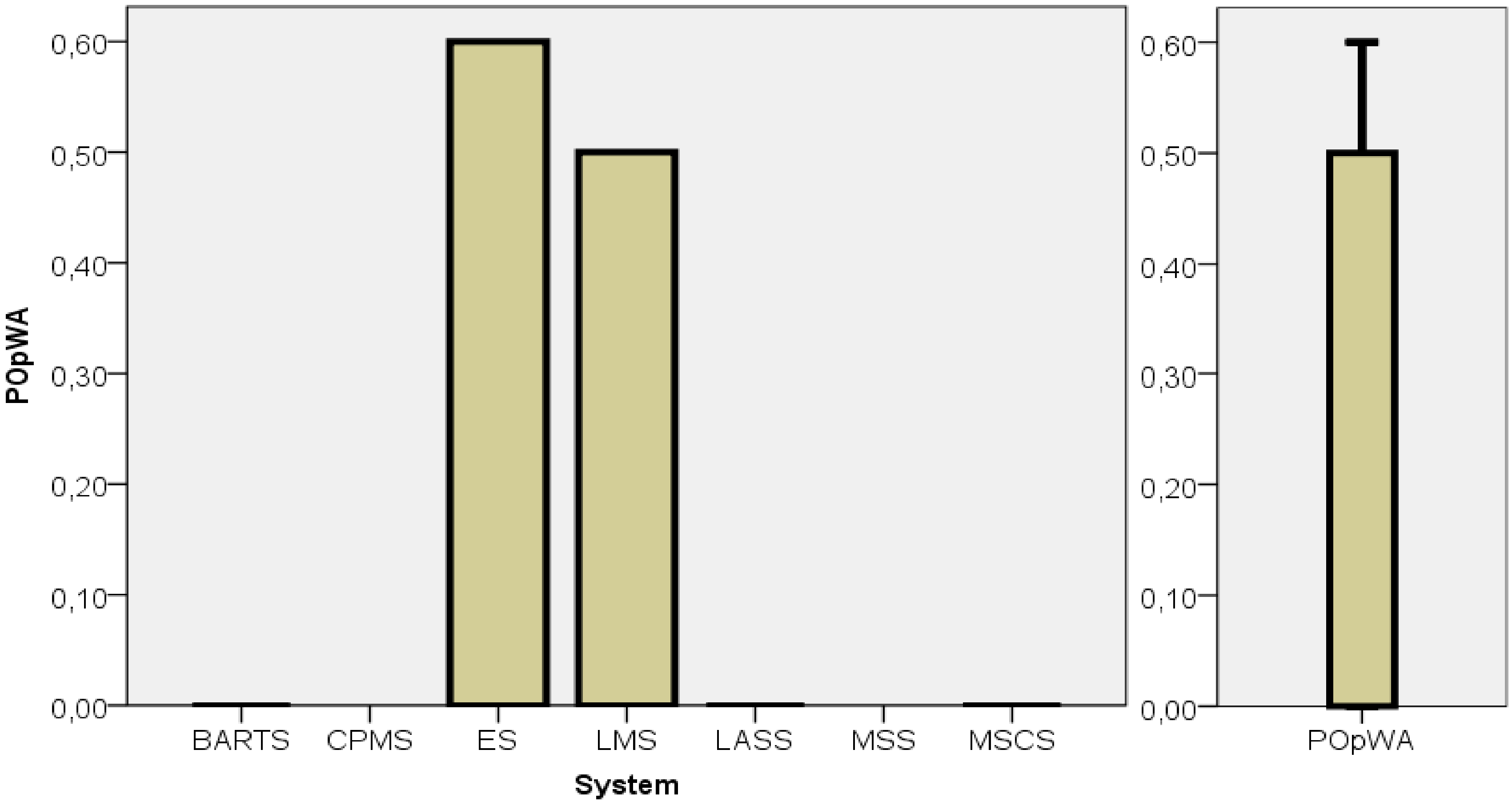
Espada, Goulão, Araújo, "*A Framework to Evaluate Complexity and Completeness of KAOS Goal Models*", CAiSE 2013, Valencia, Spain

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

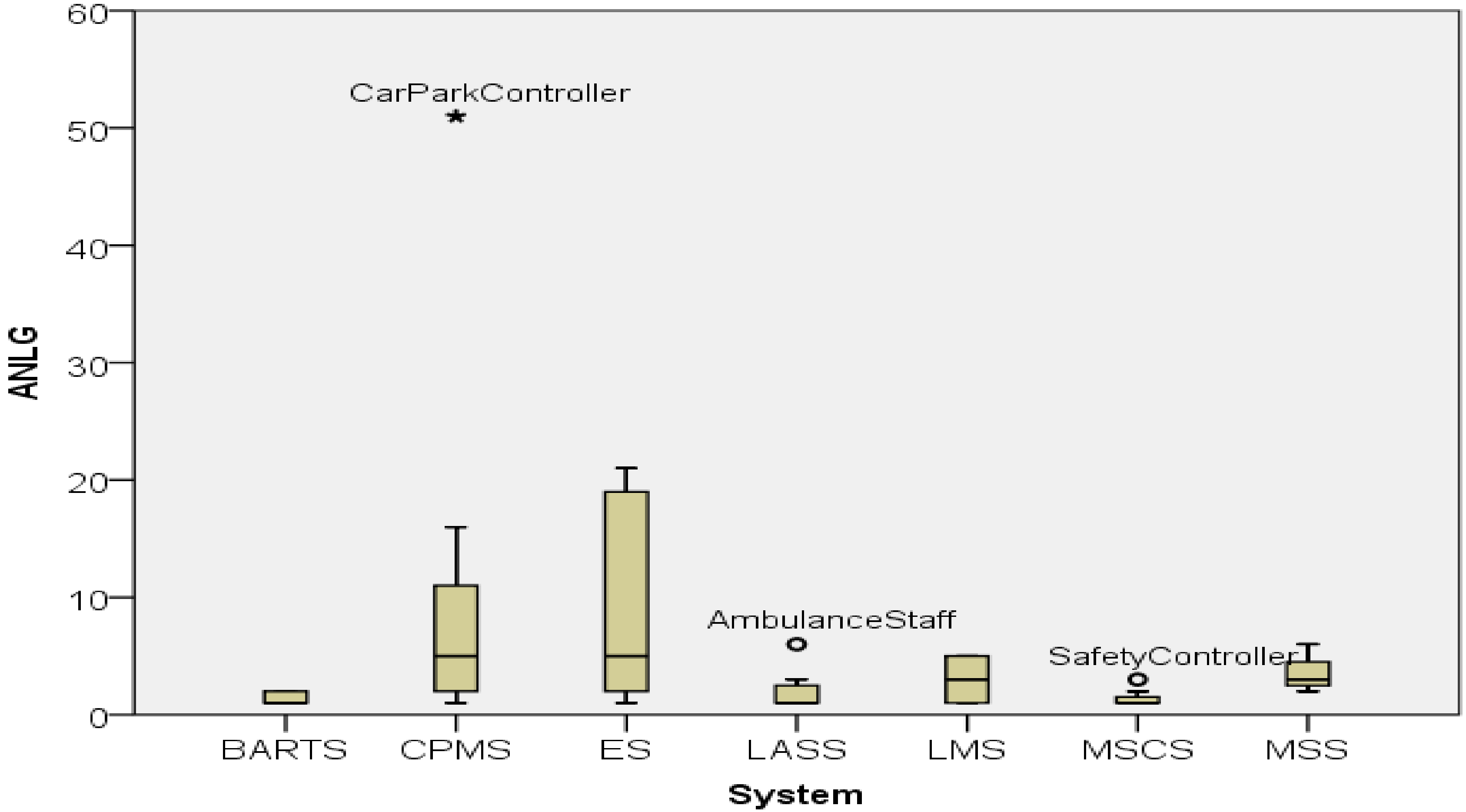
Percentage of **O**perations **w**ith an **A**gent



Espada, Goulão, Araújo, "*A Framework to Evaluate Complexity and Completeness of KAOS Goal Models*", CAiSE 2013, Valencia, Spain

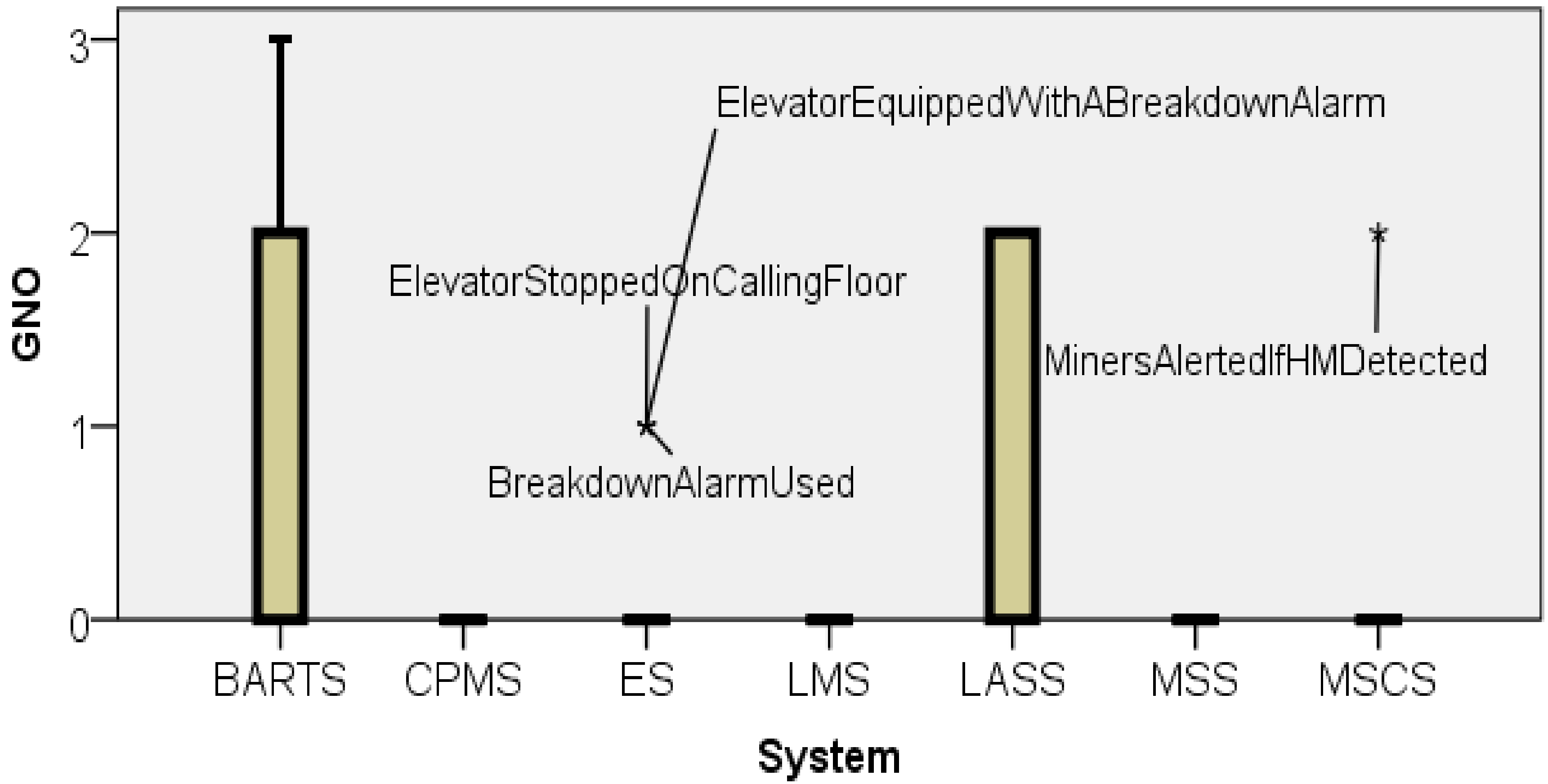
Complexity

Number of Leaf Goals per Agent



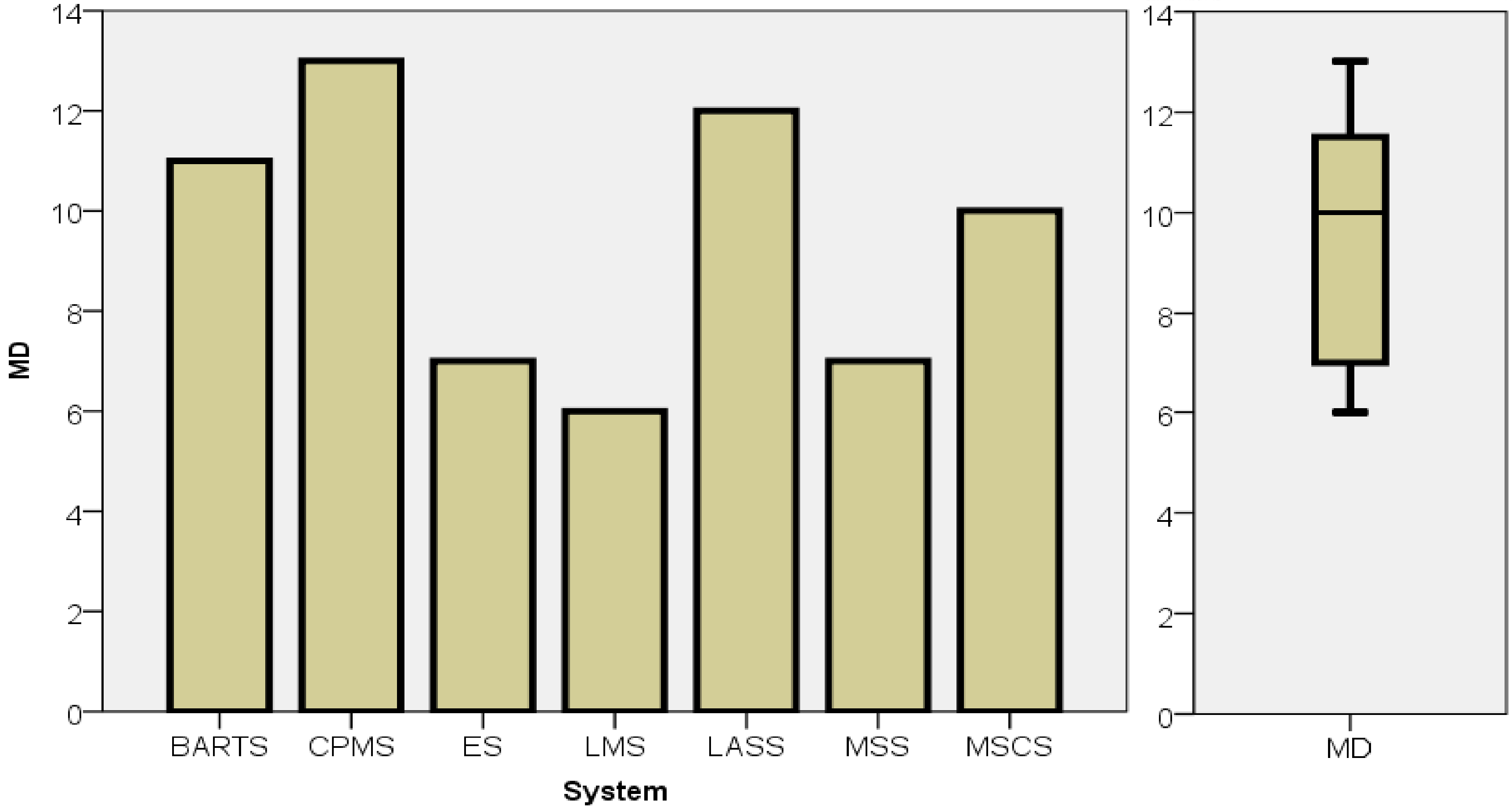
Espada, Goulão, Araújo, “*A Framework to Evaluate Complexity and Completeness of KAOS Goal Models*”, CAiSE 2013, Valencia, Spain

Objects per Goal



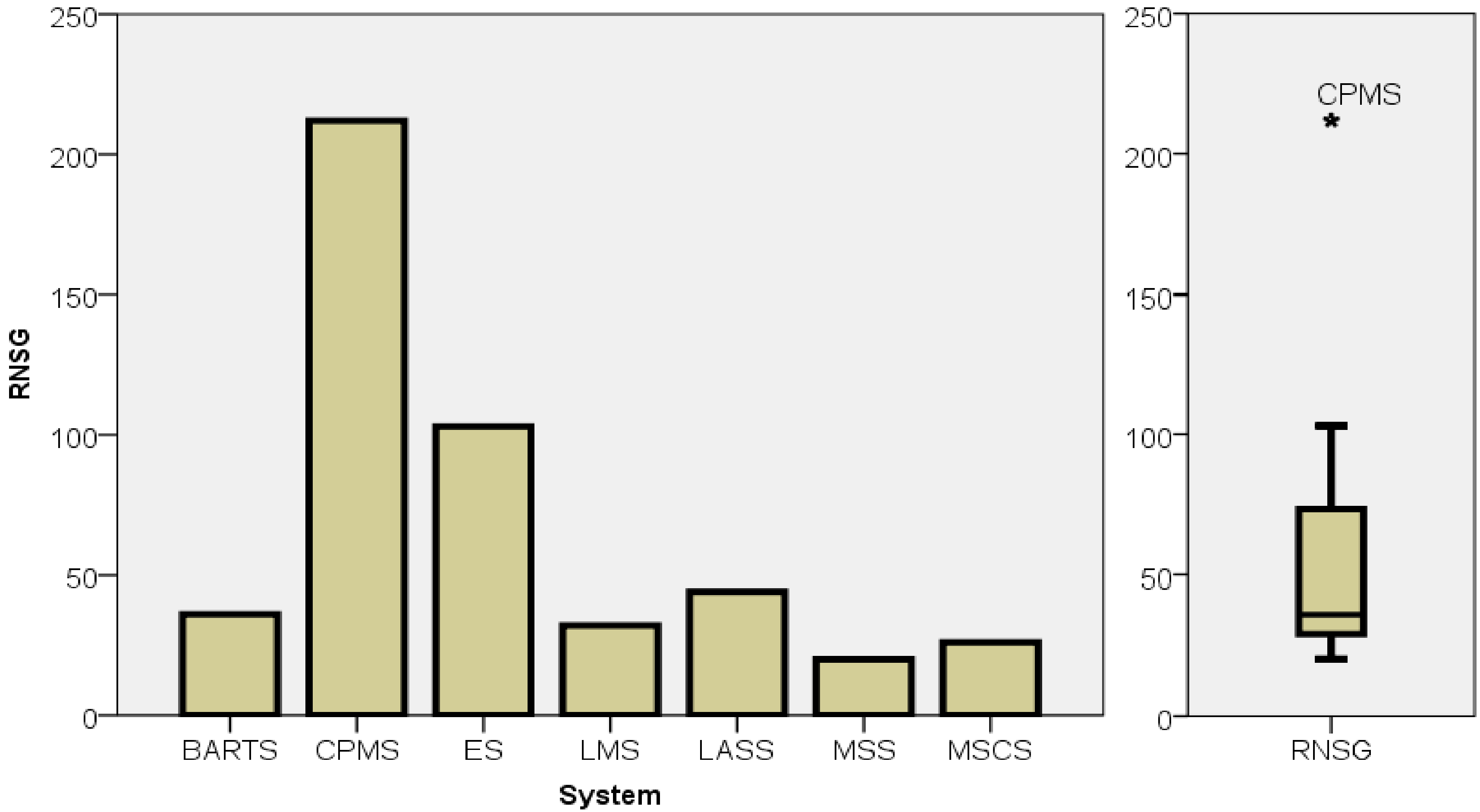
Espada, Goulão, Araújo, "*A Framework to Evaluate Complexity and Completeness of KAOS Goal Models*", CAiSE 2013, Valencia, Spain

Model Depth



Espada, Goulão, Araújo, "*A Framework to Evaluate Complexity and Completeness of KAOS Goal Models*", CAiSE 2013, Valencia, Spain

Root Number of Sub-Goals



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.

Espada, Goulão, Araújo, "*A Framework to Evaluate Complexity and Completeness of KAOS Goal Models*", CAiSE 2013, Valencia, Spain

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.

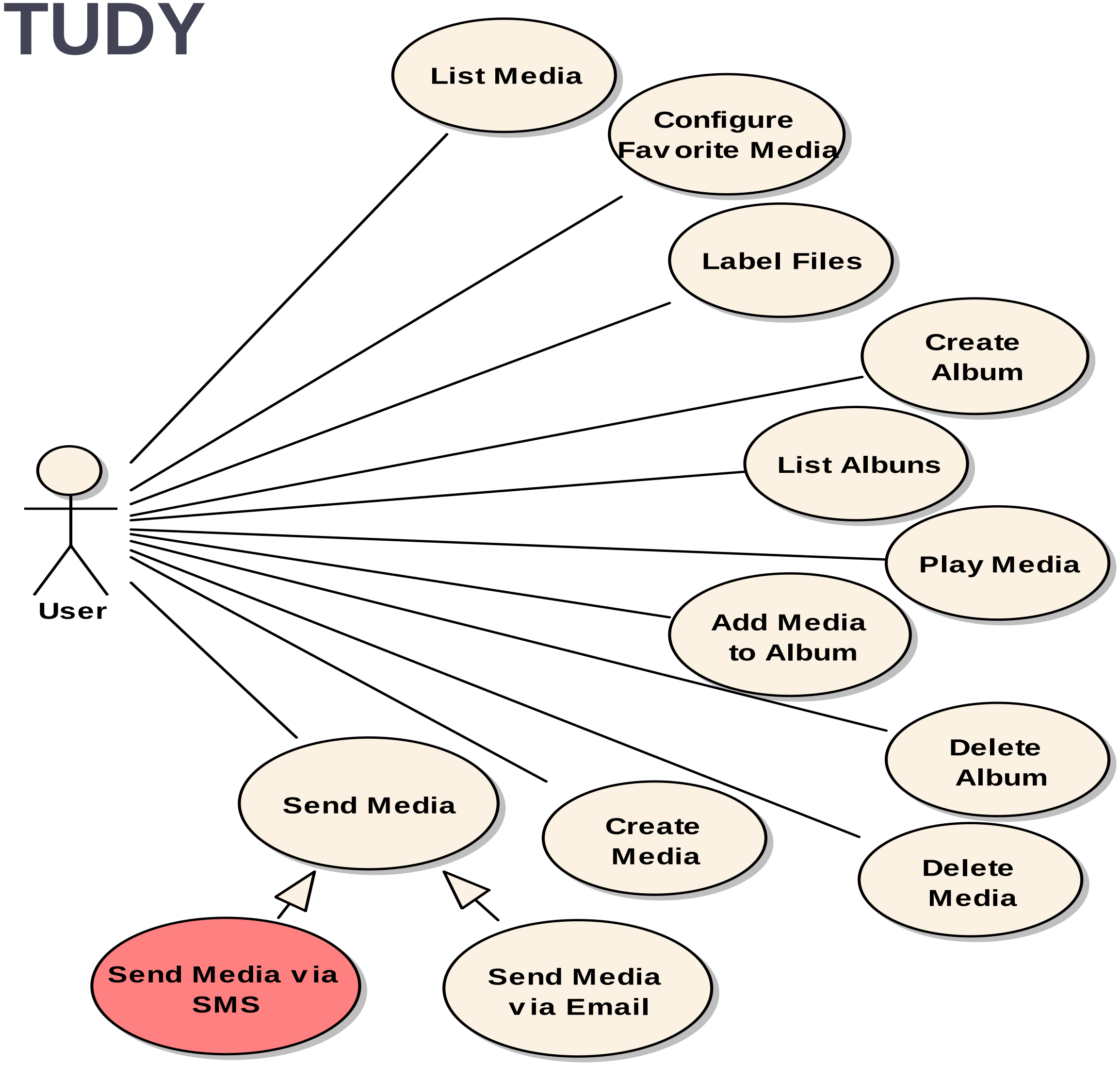
Refining Sequence Models

- 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

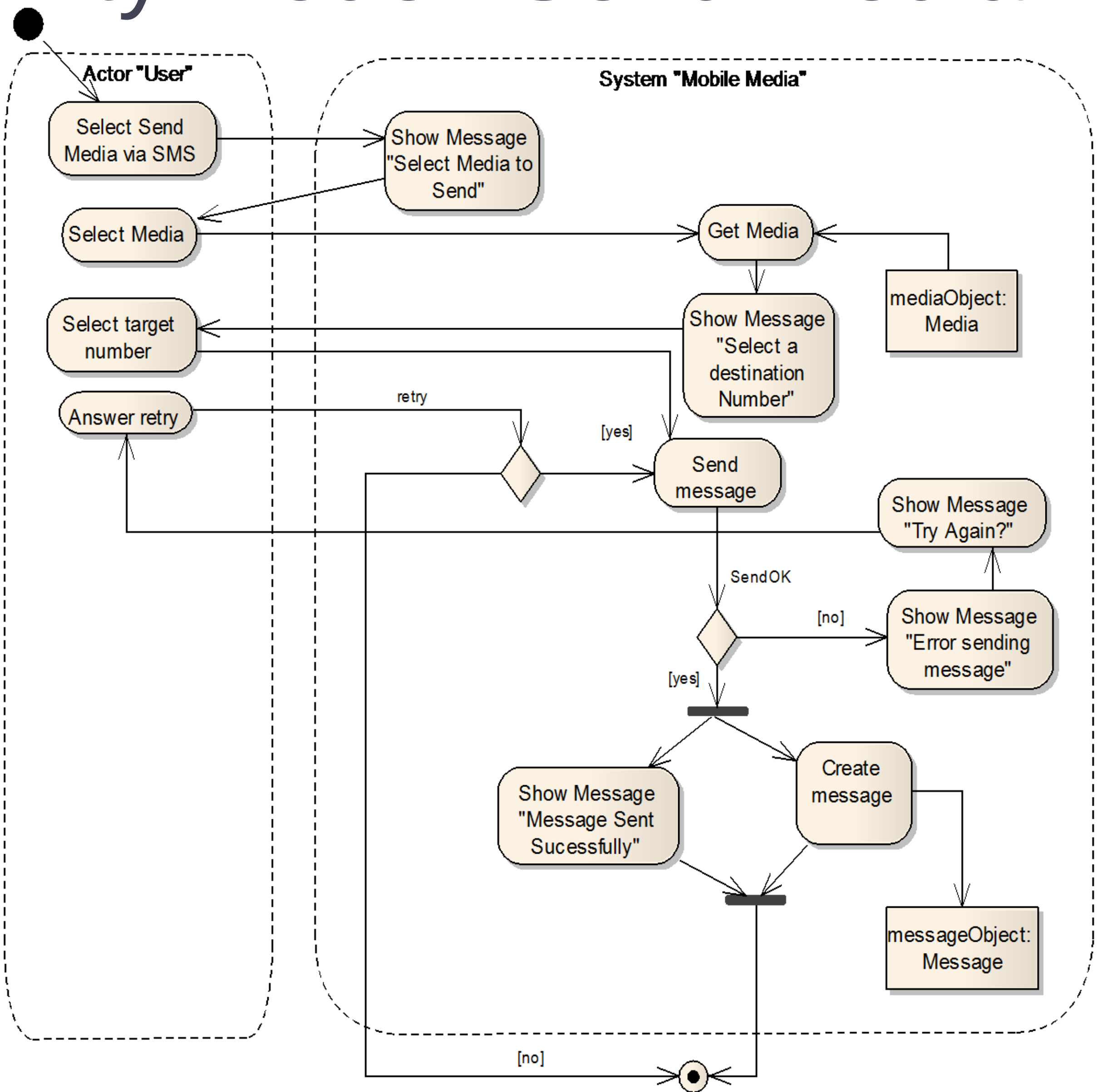
Tool support

- 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

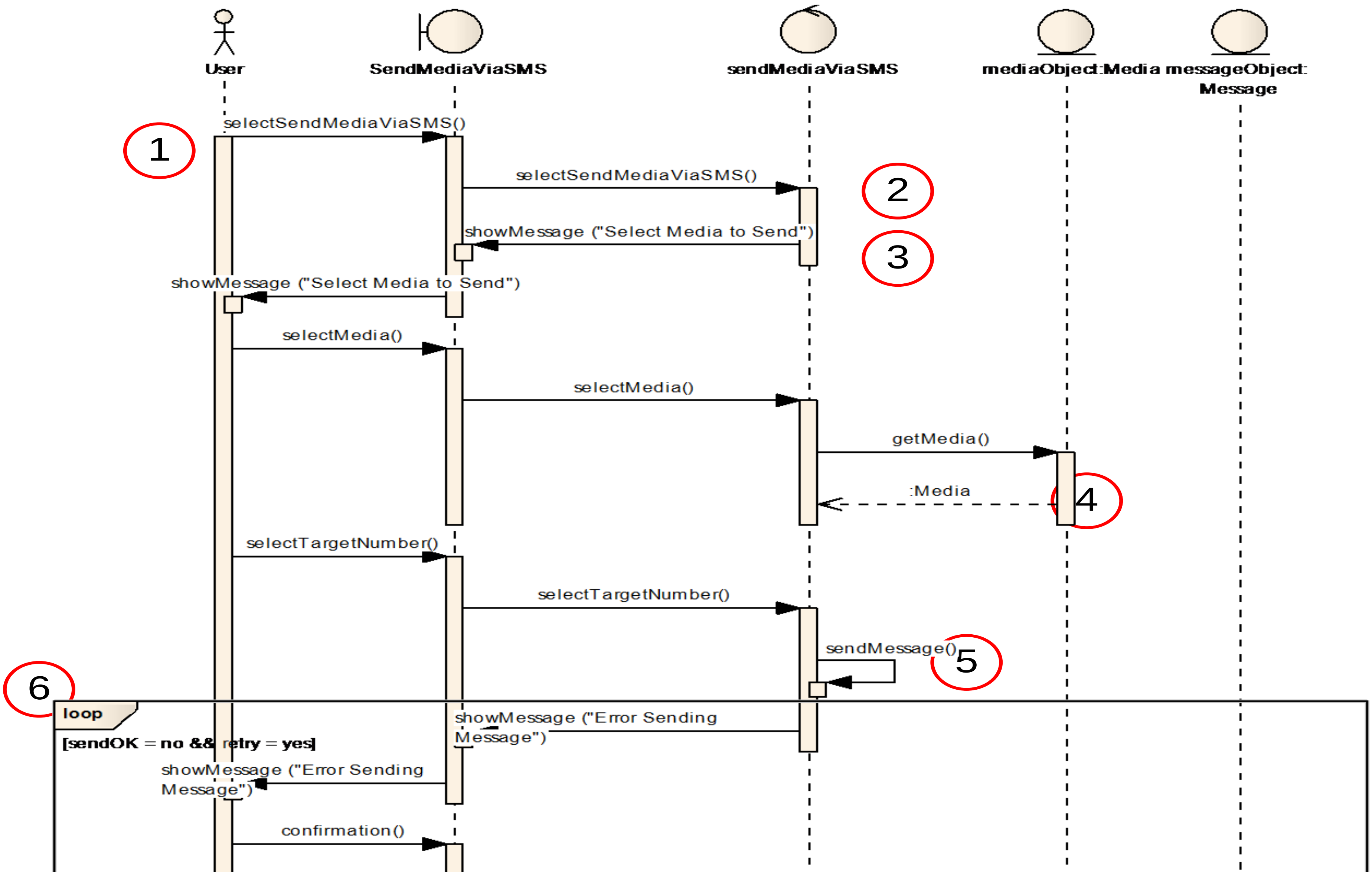
APPLICATION TO THE MOBILE MEDIA CASE STUDY

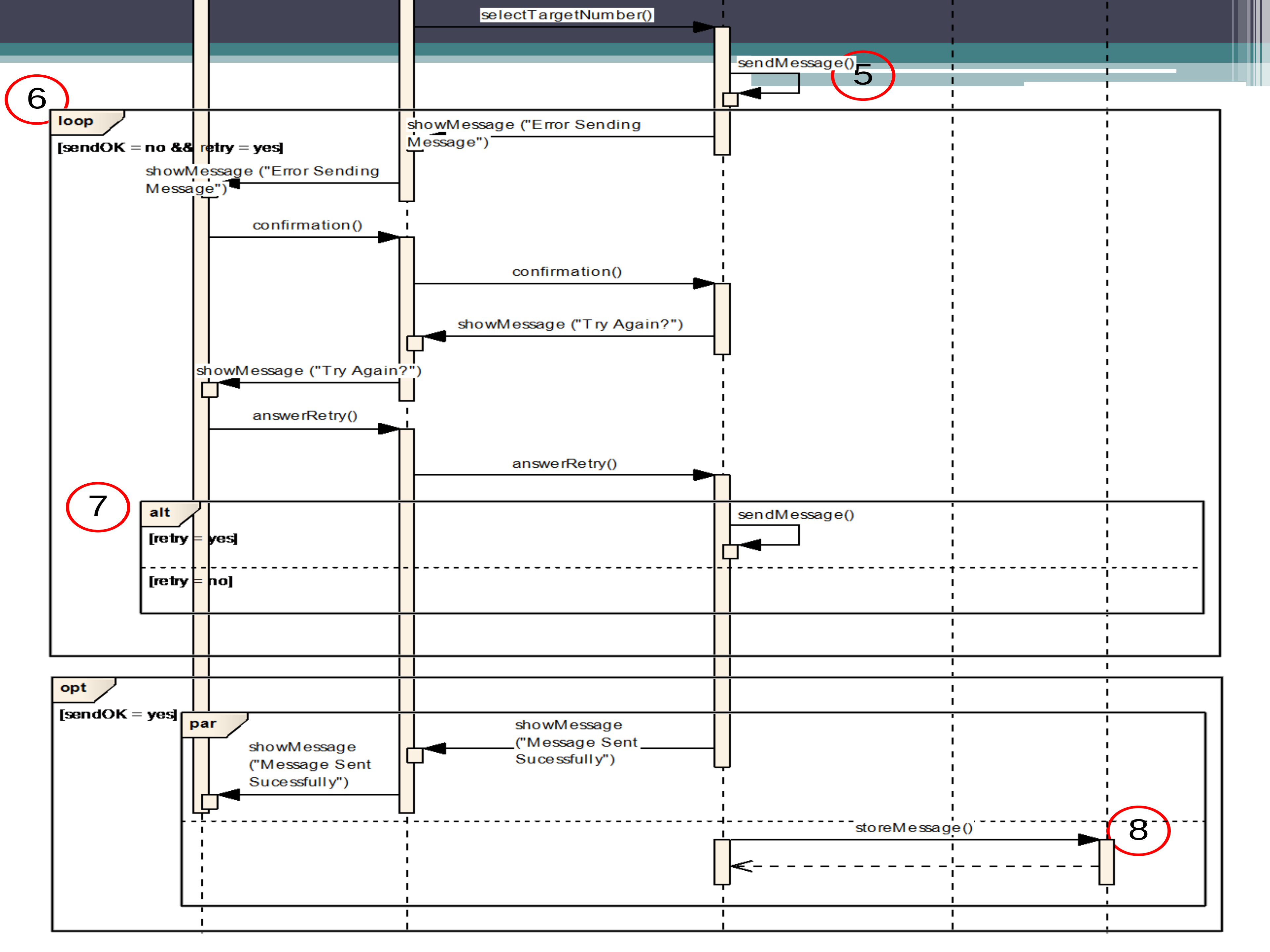


Activity model: Send Media via SMS



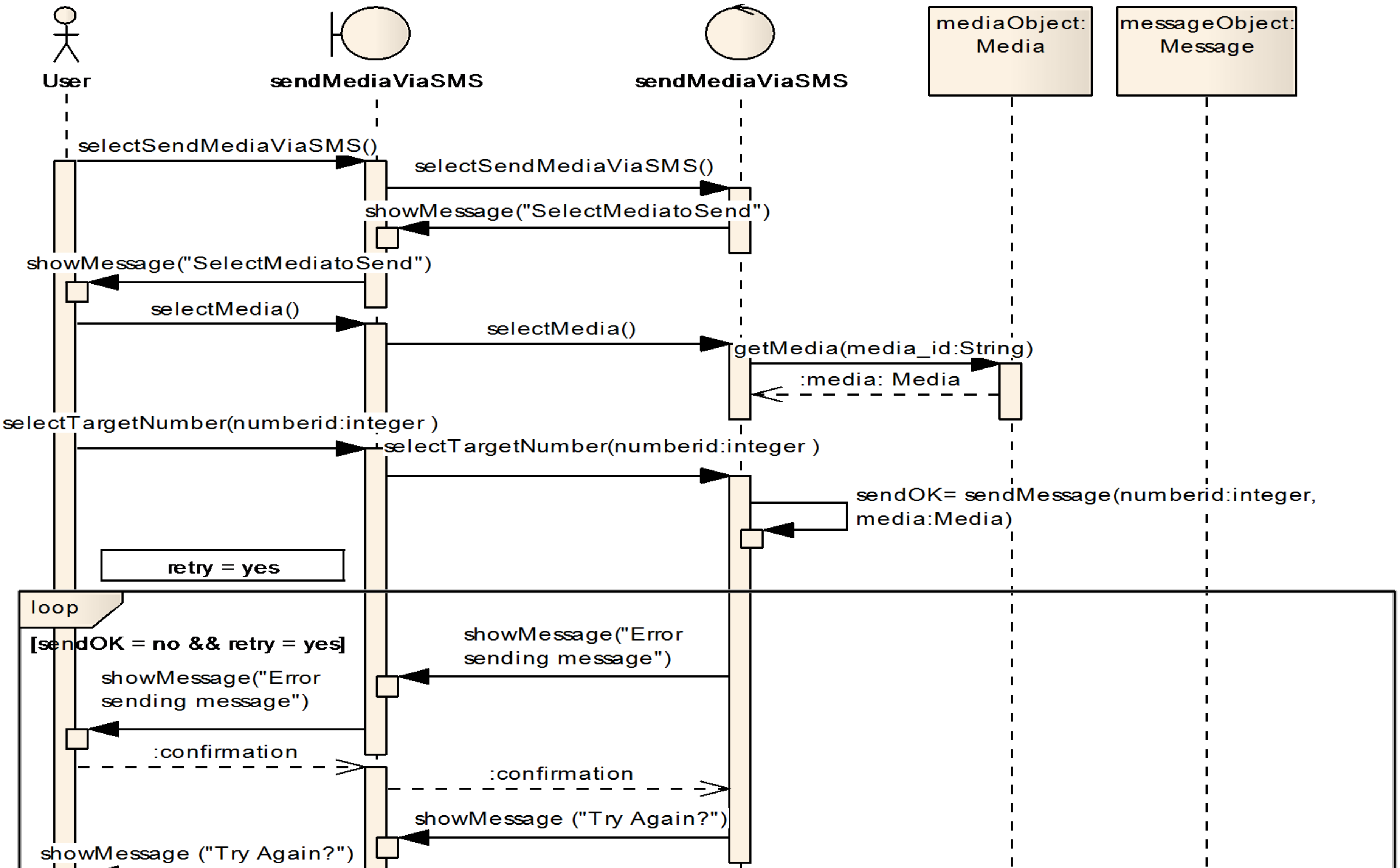
Transformation to SD

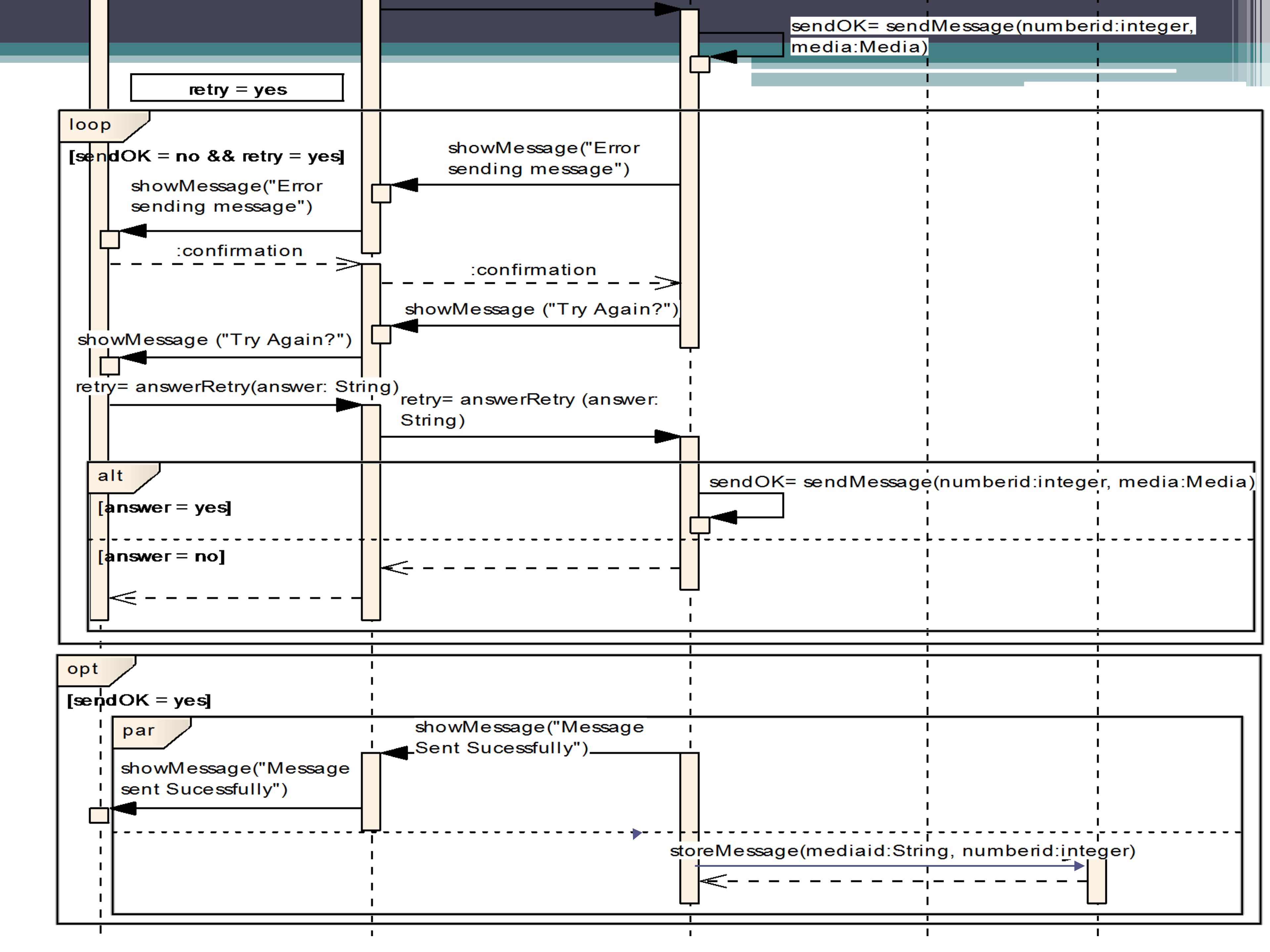




selectTargetNumber()
sendMessage()
5
6
loop
[sendOK = no && retry = yes]
showMessage ("Error Sending Message")
showMessage ("Error Sending Message")
confirmation()
confirmation()
showMessage ("Try Again?")
showMessage ("Try Again?")
answerRetry()
answerRetry()
7
alt
[retry = yes]
sendMessage()
[retry = no]
opt
[sendOK = yes]
par
showMessage ("Message Sent Sucessfully")
showMessage ("Message Sent Sucessfully")
storeMessage()
8

Refinement


User
sendMediaViaSMS
sendMediaViaSMS
mediaObject: Media
messageObject: Message
selectSendMediaViaSMS()
selectSendMediaViaSMS()
showMessage("SelectMediatoSend")
showMessage("SelectMediatoSend")
selectMedia()
selectMedia()
getMedia(media_id:String)
:media: Media
selectTargetNumber(numberid:integer)
selectTargetNumber(numberid:integer)
sendOK= sendMessage(numberid:integer, media:Media)
retry = yes
loop
[sendOK = no && retry = yes]
showMessage("Error sending message")
showMessage("Error sending message")
:confirmation
:confirmation
showMessage ("Try Again?")
showMessage ("Try Again?")



sendOK= sendMessage(numberid:integer, media:Media)
retry = yes
loop
[sendOK = no && retry = yes]
showMessage("Error sending message")
showMessage("Error sending message")
:confirmation
:confirmation
showMessage ("Try Again?")
showMessage ("Try Again?")
retry= answerRetry(answer: String)
retry= answerRetry (answer: String)
alt
sendOK= sendMessage(numberid:integer, media:Media)
[answer = yes]
[answer = no]
opt
[sendOK = yes]
par
showMessage("Message Sent Sucessfully")
showMessage("Message sent Sucessfully")
storeMessage(mediaid:String, numberid:integer)

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.

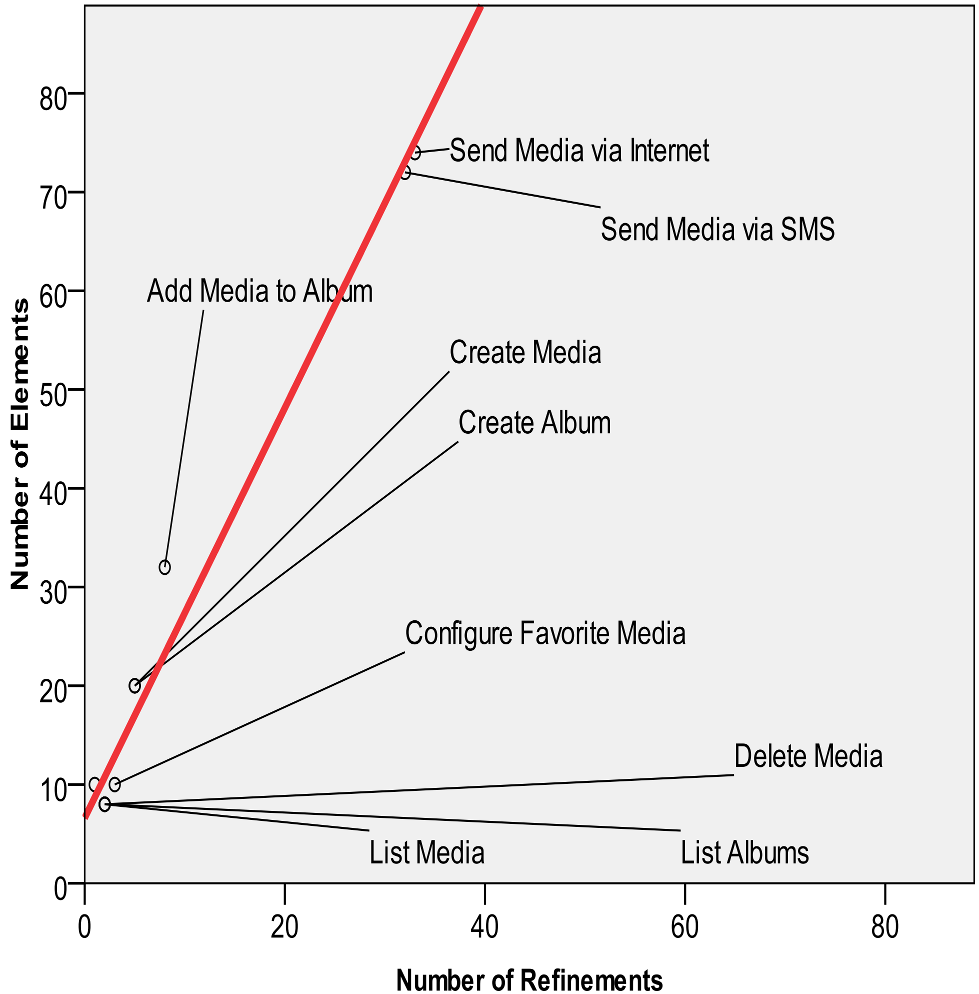
Insertions and deletions

- 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.

NUMBER OF ELEMENTS AND REFINEMENTS FOR EACH SCENARIO

Scenarios	**Number of Elements**	Number of Insertions	Number of Removals	**Number of Refinements**
Send Media via SMS	**72**	30	2	**32**
Send Media via Internet	**74**	31	2	**33**
Create Media	**20**	4	1	**5**
Delete Media	**8**	2	0	**2**
Create Album	**20**	4	1	**5**
Add Media to Album	**32**	6	2	**8**
List Media	**8**	2	0	**2**
Configure Favourite Media	**10**	2	1	**3**
List Albums	**8**	2	0	**2**
Play Media	**10**	3	0	**3**
Delete Album	**8**	2	0	**2**
Total	**270**	88	9	**97**

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?