Building
Effective Modelling Domains

Sagar Sen
Benoit Baudry
Jean-Marc Jezequel

Outline

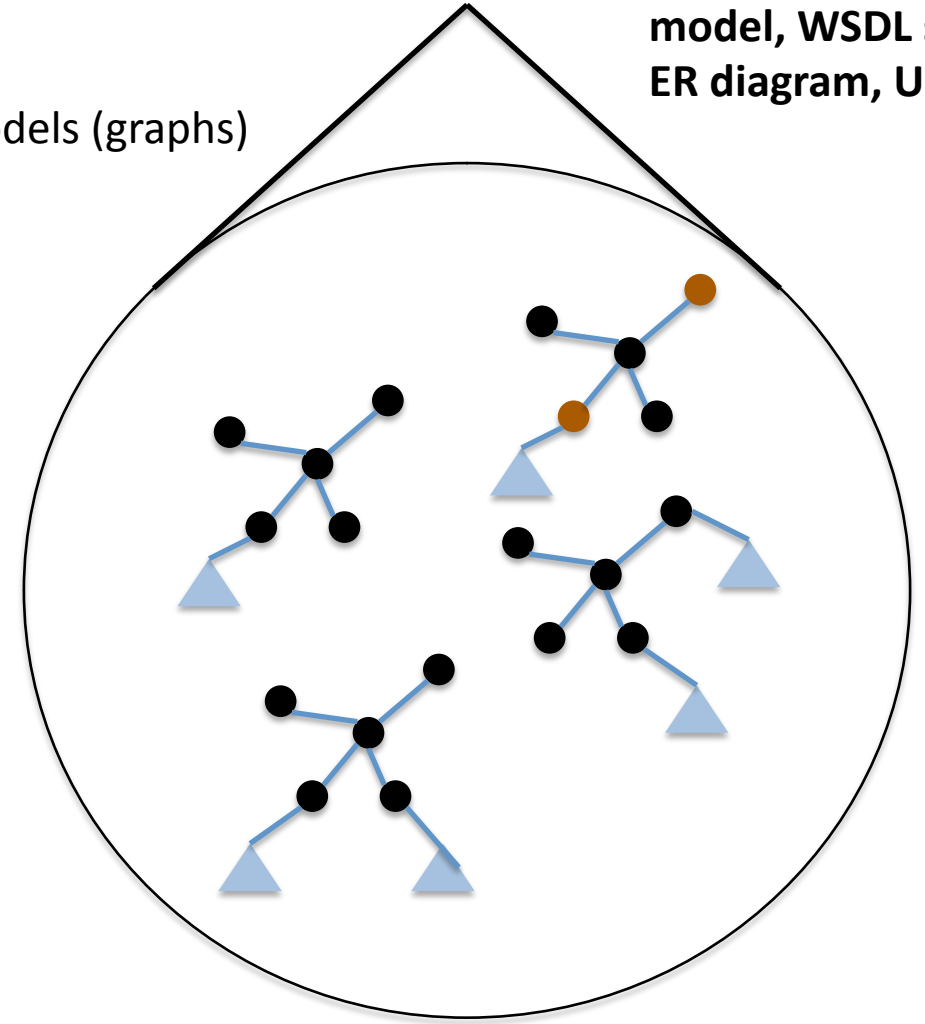
- What is a Modelling Domain ?
- What is an Effective Modelling Domain ?
- Specifying an Effective Modelling Domain
- The Problem : Generation of Models in the Effective Modelling Domain
- The Solution and Demo
- Case Study
- Conclusion and Future Work

What is a Modelling Domain ?

Meta-model

in a meta-meta language ECore
model, WSDL specification,
ER diagram, UML Class Diagram etc.

specifies a set of models (graphs)



Effective Modelling Domains

- Merriam-Webster Dictionary definition of the term Effective:

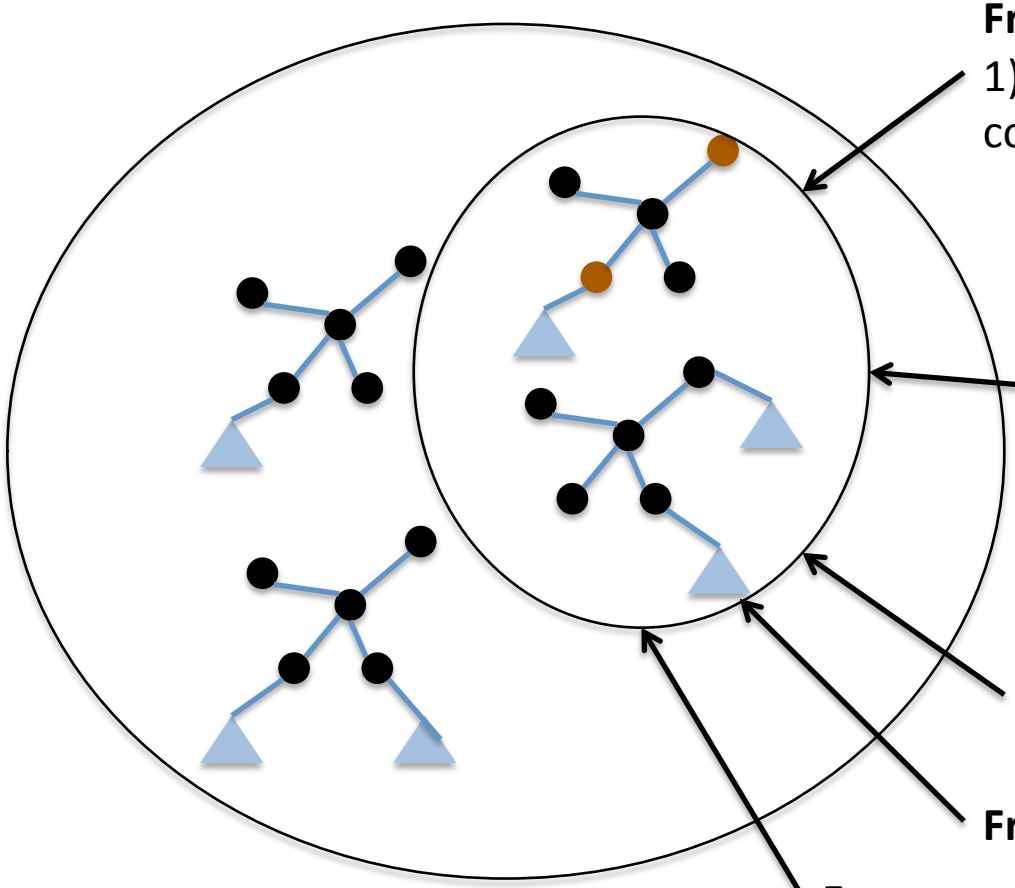
"Producing a decided, decisive, or desired effect"

Therefore, we define an effective modelling domain as:

A modelling sub-domain containing models with characteristics that produce desired effects

Eg.: Models that can detect bugs in a model transformation. Models of WSDL specifications that can test service weaving or obtain accurate QoS measures.

Specifying an Effective Modelling Domain



Knowledge that define an effective modelling domain

From the meta-modelling world…
1) Meta-model invariants in a constraint language such as OCL

From the model transformation world…
1) Model Transformation pre-condition
2) Mutation analysis scores
3) Test Criteria
4) Behavioral Traces

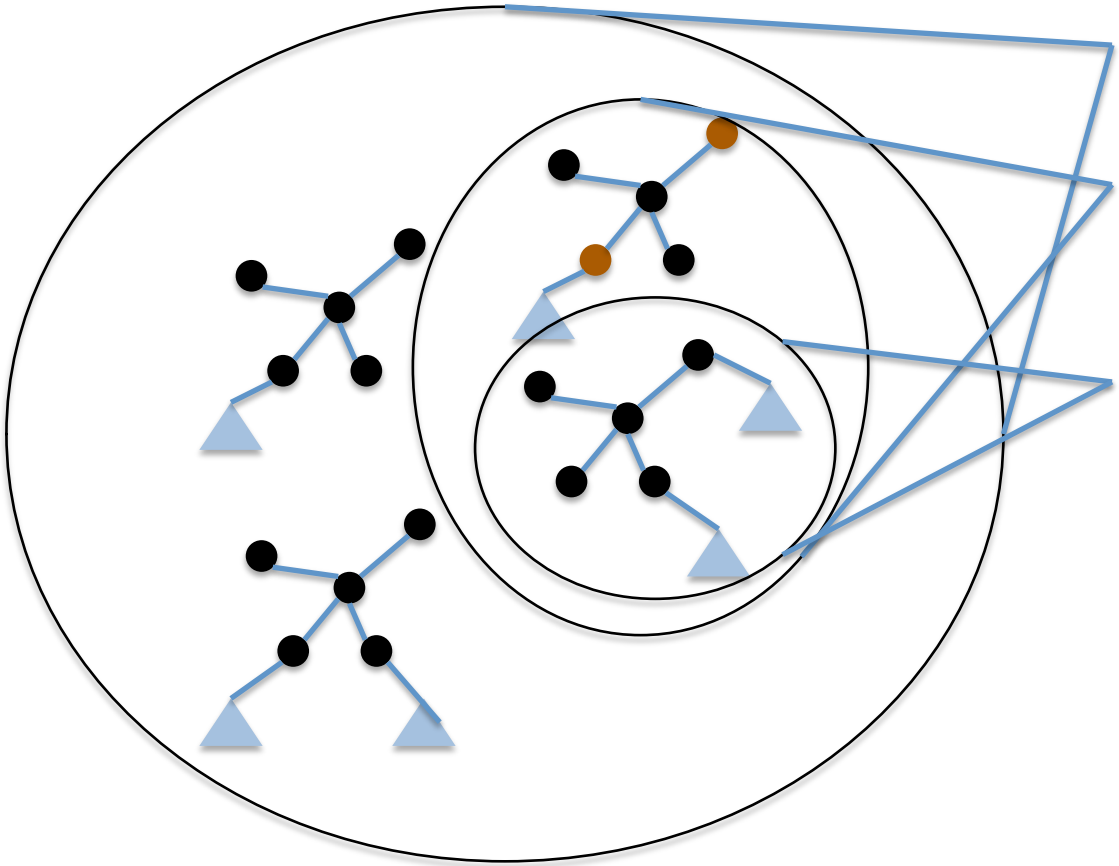
From a domain-specific world such as Web Services…
1) QoS

From a partial model

From an experimental design

The need : Models in the effective modelling domain

The Problem (1)



Modelling Domain

Actual Effective Modelling Domain

Effective Modelling Domain with Finite-scope Automatic Model Synthesis

Question we want to answer :

How do we obtain a sub-domain of the effective modelling domain with support for Finite-scope Automatic Model Synthesis ?

The Problem (2)

Challenges

Conceptual

1) Generated models must entail knowledge from modelling domain and effective modelling domain
2) If the effective modelling domain is satisfiable we must be able to generate more than one model (if they exists)
3) We must be able to generate models of different sizes and with different properties

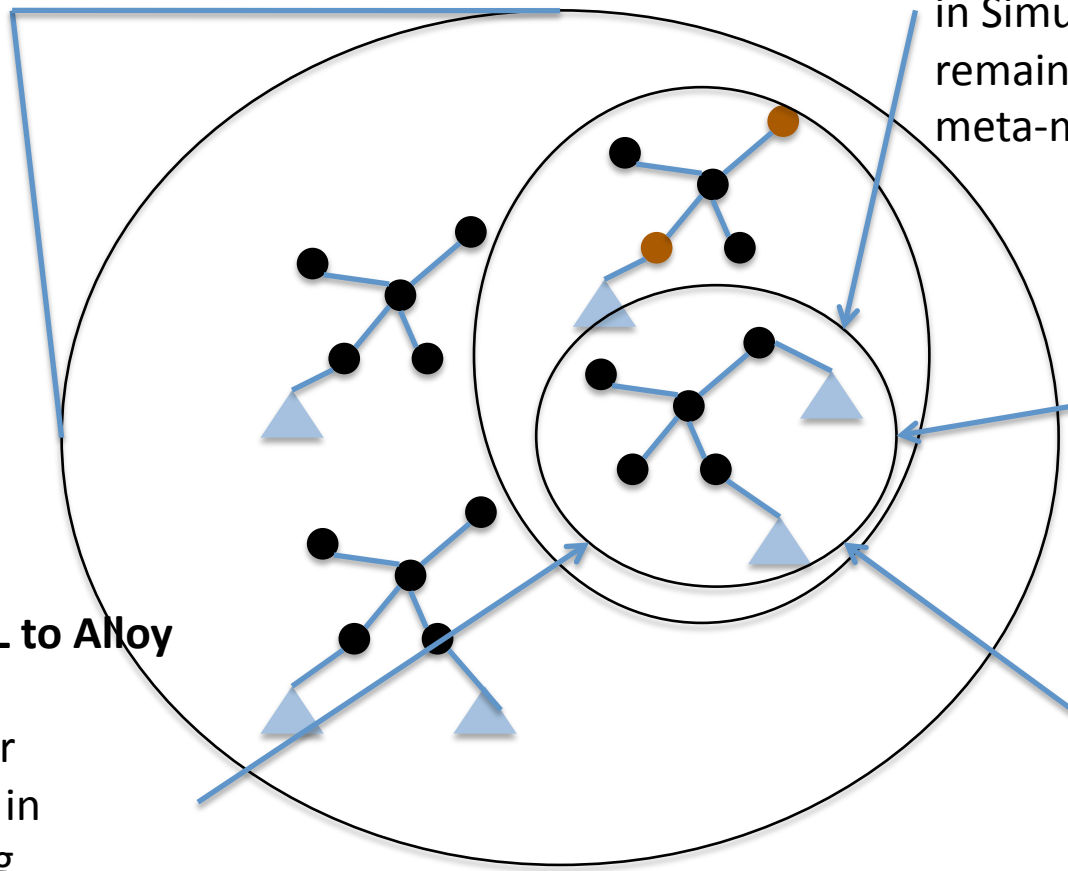
Technical

1) Generated models must be output in the XML format of the modelling domain. Eg. XMI for ECore.
2) We must be able to integrate/combine knowledge (that define the effective modelling domain) in different formats/languages/files and developed at different times.

Solution : Cartier

Modelling Domain
Ecore2Alloy
(*Status*: Automated using XSLT)

Partial Model to Alloy Predicates
(*Status*: Automation presented in Simulation paper, questions remain about meta-model)



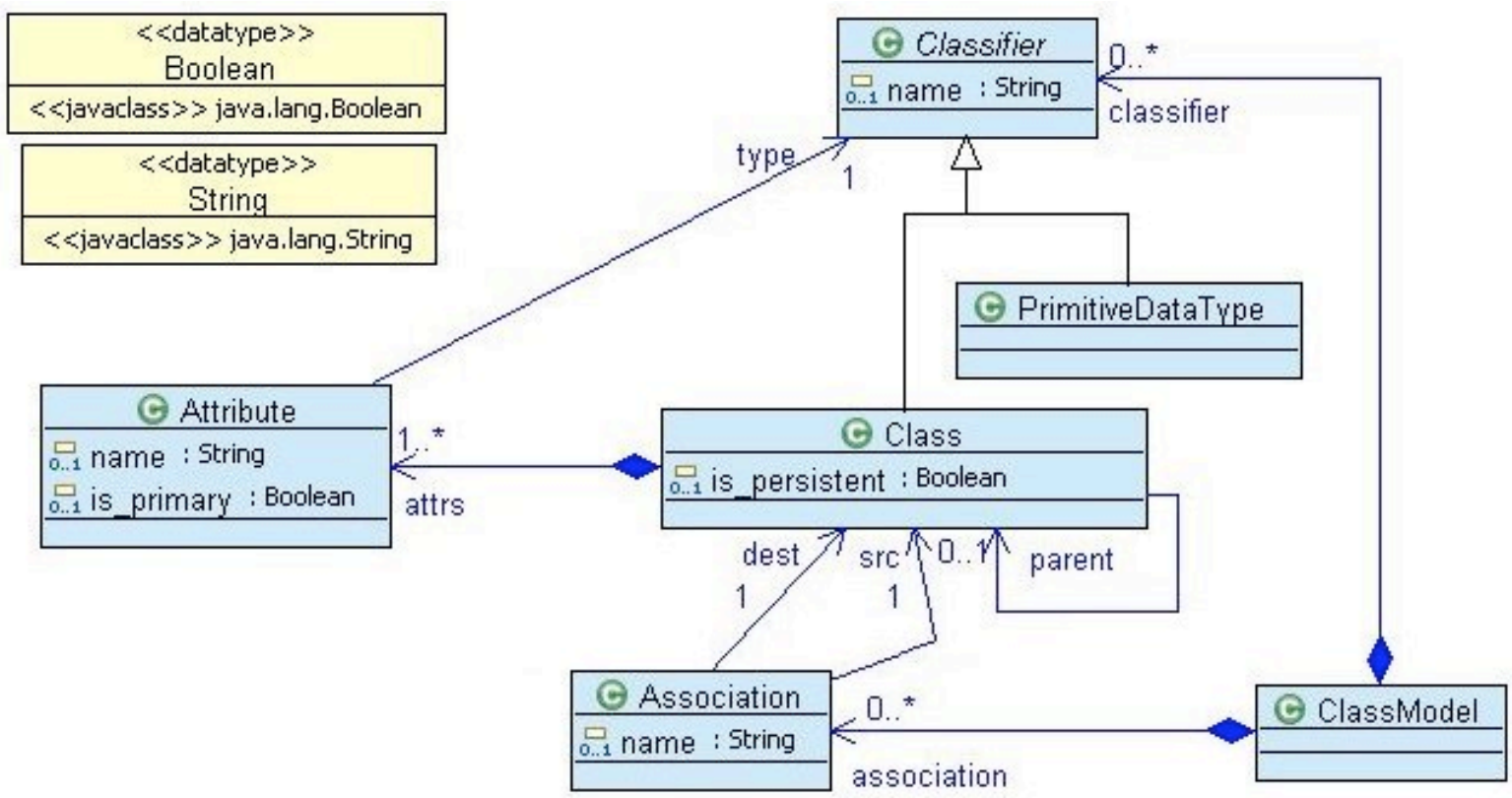
Experimental Design to Alloy Run Statements
(*Status*: Implemented using a Python script)

Subset of OCL to Alloy
OCL2Alloy
(Status: Under development in Kermeta using Olivier's work on OCL2KMT) :

Model Fragments to Alloy Predicates
(*Status*: Implemented by Benoit et al. in MMCC + extension)

(OCL is used to specify meta-model invariants, pre-conditions etc.)

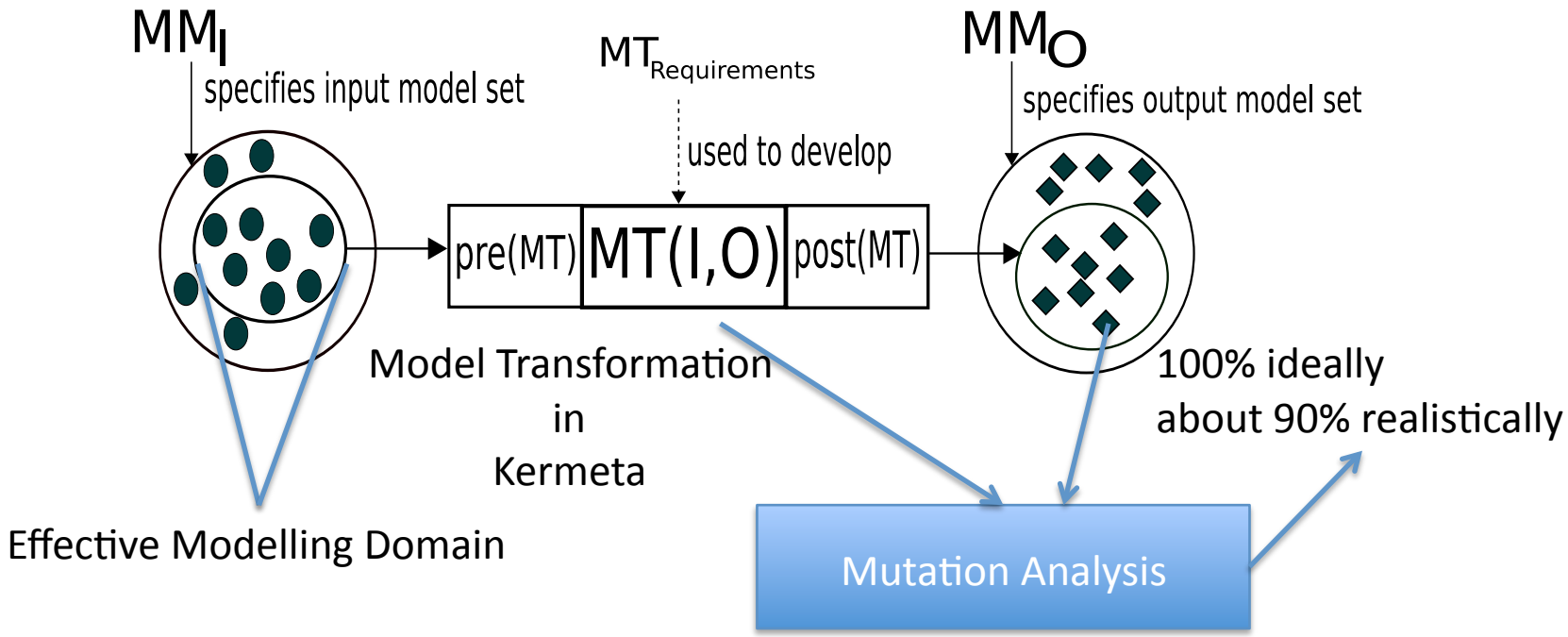
Demo : Simple UMLCD to Alloy



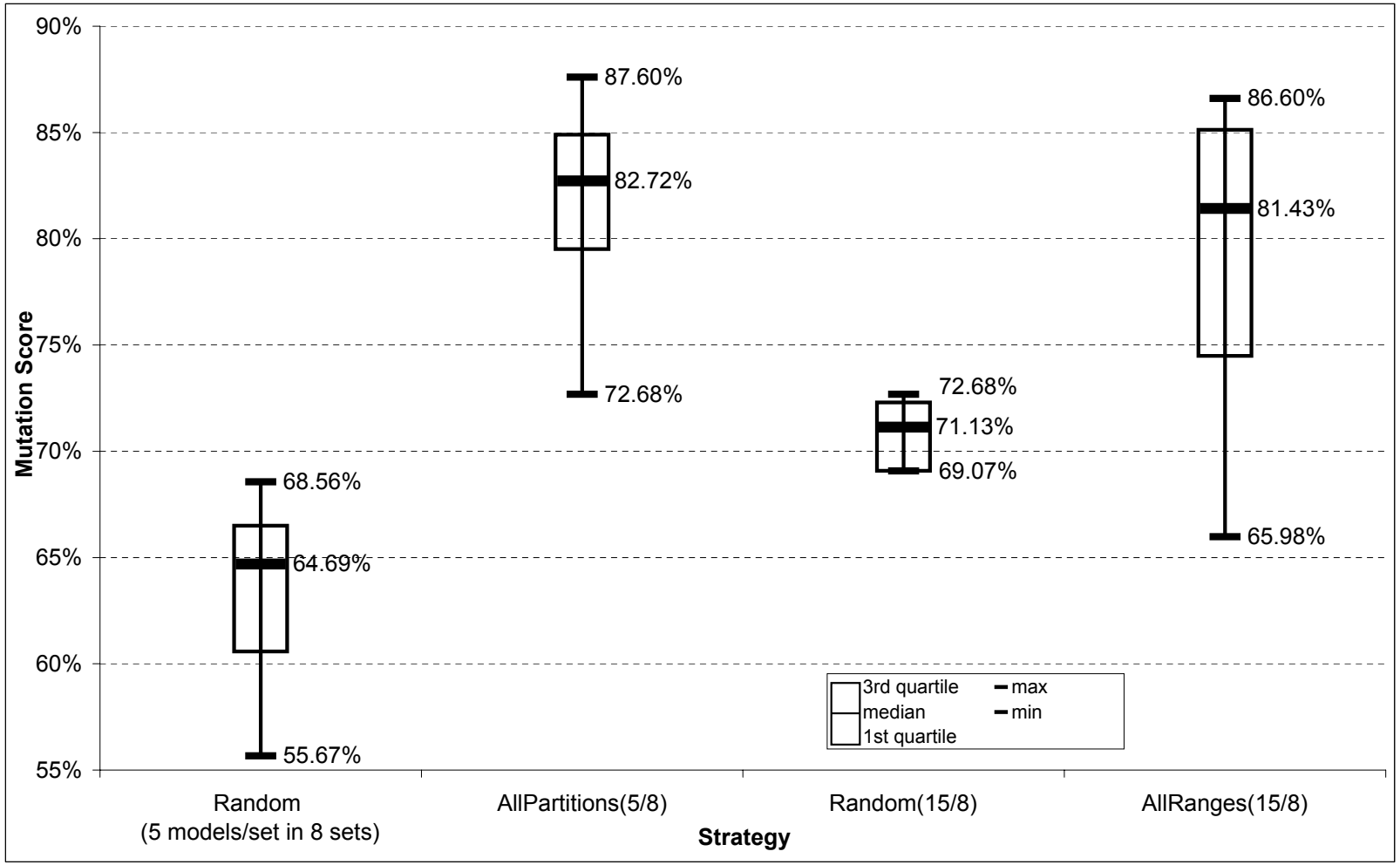
Input : simpleuml_mm.ecore

Case Study: Model Transformation Testing

Effective Modelling Domain:
Set of test models in the input domain that can detect bugs in a *model transformation*



Case Study: Recent Results based on 360 models and 50 hours of Computation



Conclusions and Future Work

1. We introduce the tool Cartier used to transform knowledge form various sources to Alloy (a common constraint language)
2. We show how we used Alloy to synthesize several hundred models to test model transformations
3. We show that Model Fragments help obtaining an effective modelling domain capable of detecting bugs in a transformation.

Now, I am funded by a new project S-Cube…so what next…

1. Testing Web Services. Generating models conforming to WSDL specifications
2. Verifying a Web Service's operation against QoS
3. Possibility to collaborate with DiVa folks on adapting composite web services to satisfy QoS measures.
4. General ideas about how to optimize combination of knowledge from different sources?
5. Using the notion of *maximum common subgraph* to intuitively decide which part of an input model is responsible for a certain behavior.