

Efficient Software Engineering with Formal MDA

Sudhir Agarwal, Max Völkel

Institute of Applied Informatics and Formal Description
Methods (AIFB),
University of Karlsruhe, Germany

Semantic Web enabled Software Engineering
November 2005, Galway, Ireland

Introduction

- The idea of MDA is great
 - MDA suggests stepwise refinement of models
 - Graphical models are better suited for communication among humans than code
 - Models can be used to generate code
 - relieves programmers from many trivial tasks
 - enables rapid prototyping of ideas
 - ...

Shortcomings of MDA

- Developing models is costly. In practice,
 - role of UML, ER models is often confined to
 - communication among domain experts
 - code generation; once the code is available, programmers hardly use the models anymore
- Translation loses some important features,
 - e.g. relationships as first class citizens and cardinality constraints
 - each translation tool can produce different code from same model → dependency on the tool
- Modeling 100% manually; No reasoning support
 - e.g. for finding inconsistencies and redundancies in the model
- Lack of support for dynamic aspects

Formal MDA

- Two most important features of formal descriptions are
 - Automatic reasoning support
 - Conceptual as well as executable models

- How can these features improve various software engineering phases?
 - requirement specification,
 - analysis
 - design,
 - implementation,
 - testing
 - ...

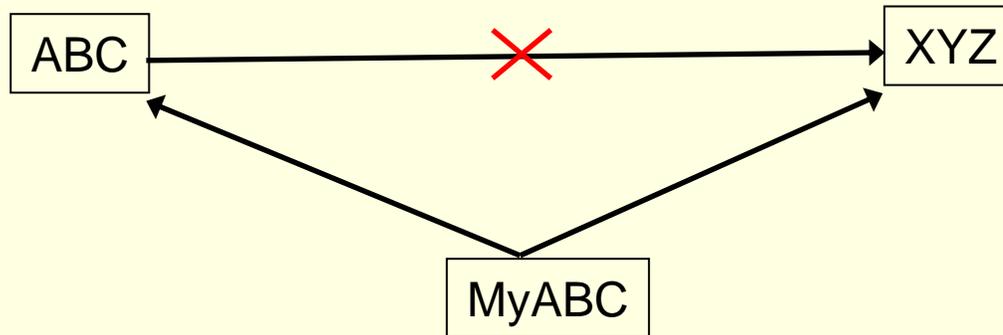
Req. Specification and Analysis

- Requirements can be specified more precisely
 - enables identifying problems and open issues early
 - reduces misinterpretation by programmers
- Reasoners can be used in the analysis phase
 - to find inconsistencies and redundant artifacts automatically
- Reduce communication gap
 - reduce semantic errors in the software

Design / Modularity

- Ontologies make schema integration easier
 - relationships are first class citizens

ABC should become sub-class of XYZ



Implementation and Testing

- Models as software assets
 - reuse and search for already modeled artifacts
- No translation needed since conceptual models are executable at the same time
 - allows different views on the same model
- Test cases (e.g. JUnit) may be generated automatically from the requirement specification
 - automatic testing of the programmed software

Conclusion

- UML, ER and XML popular but don't offer much reasoning support
- Formal MDA is less popular but offer added value due to formal semantics and reasoning support
- Formal MDA may require more effort in the early stages of SE, but can be more efficient at the end
- Is making ontologies more popular harder than adding semantics to UML?

Thank You!