Understand complex changes and improve the quality of your UML and domain-specific models

Thorsten Arendt, Timo Kehrer, Gabriele Taentzer

Philipps-Universität Marburg,
Universität Siegen,
Germany
Content

- Introduction
- Part I: Analyze and improve UML models
  - Quality assurance of UML models using
    - model metrics, smells, and refactorings
  - Understanding complex model changes by
    - recognizing edit operations

Break

- Part II: Analyze and improve DSL models
  - Example DSL for simple Web applications
  - Specifying and applying new QA techniques
  - Specifying edit operations to understand model changes

- Conclusion

Understand complex changes and improve the quality of your models
Introduction
Motivation: model quality assurance

Understand complex changes and improve the quality of your models
Motivation: model quality assurance

Understand complex changes and improve the quality of your models
Understand model changes

Do you see the differences?

Understand complex changes and improve the quality of your models
Model versions and variants

Understand complex changes and improve the quality of your models
Understanding model evolution

Understand complex changes and improve the quality of your models
Collaborative work and 3-way merging

Understand complex changes and improve the quality of your models
Propagation of changes and model patching

Understand complex changes and improve the quality of your models
Part I: Analyze and improve UML models
A model quality assurance process
A model quality assurance process

Specification of a project-specific model quality assurance process
A model quality assurance process

Application of the specified process to concrete software models
Model quality aspects: 6C goals

- Correctness
  - correct wrt. language
  - correct wrt. domain

- Completeness
  - all relevant information
  - detailed enough wrt. purpose

- Consistency
  - no contradictions
  - horizontally/vertically

- Comprehensibility
  - understandable by intended users
  - aesthetic and simple

- Confinement
  - suited to modeling purpose
  - relevant information on the right abstraction level

- Changeability
  - can be evolved rapidly

---

Scenario: vehicle rental company

- **Project**
  - Development of a web application for renting vehicles

- **Semantic domain**
  - Company with vehicles, customers, services, ..

- **Technical domain**
  - Web application

- **Modeling purpose**
  - Modeling of the semantic domain as part of the analysis

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Scenario: initial UML model

- Confinement
  - suited to modeling purpose
  - relevant information on the right abstraction level

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Demo
Metrics configuration dialog

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Metrics results view

Understand complex changes and improve the quality of your models

Arendt, Kehrer, Taentzer
Metrics report dialog

EMF Quality Assurance - Metrics Reporting

Please insert the required informations for reporting your metrics results.

Please select the output format(s) of your metrics report.

- [x] XML (default)
- [ ] HTML
- [ ] PDF
- [ ] PS
- [ ] DOC
- [ ] PPT
- [ ] XLS
- [ ] ODP
- [ ] ODS
- [ ] ODT

Please select either a predefined report design from the list or import your custom design from the file system.

- TUBErptdesign

Please choose location and name of your metrics report.

D:/MoDELS2013/workspace/umlmodel.example/reports/metrics_report

Finish  Cancel
Metrics report: HTML + tube diagram

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Smells configuration dialog

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Smells results view

Highlighting of involved elements

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Refactorings as quick fixes

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Partially improved model

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Improved model

Understand complex changes and improve the quality of your models

Arendt, Kehrer, Taentzer
Textual comparison

Understand complex changes and improve the quality of your models

Arendt, Kehrer, Taentzer
Model differencing: State-of-the-Art

Understand complex changes and improve the quality of your models
Generic difference engine

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Low-level difference

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
EMF Compare: UML diff engine

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Edit operation detection with SiLift

Model A

Model B

Matching

Difference Derivation

Low-level Changes

Edit Operation Detection

Edit Operations

Understand complex changes and improve the quality of your models
Operation detection: settings

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
SiLift: Detected Edit Operations

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models.
Grouping of low-level changes

Understand complex changes and improve the quality of your models

Arendt, Kehrer, Taentzer
Part II: Analyze and improve DSL models
Scenario: vehicle rental company

- **Project**
  - Development of a web application for renting vehicles using MDD

- **Semantic domain**
  - Company with vehicles, customers, services, ..

- **Technical domain**
  - Web application

- **Modeling purpose**
  - Modeling of the semantic domain as well as the information mandatory for code generation using a DSML

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Example: a DSL for simple Web applications

Understand complex changes and improve the quality of your models
Model-driven development

Models are primary artefacts and need specific tooling:
- specific model editors and code generators
- specific quality assurance tools
- specific model differencing

Understand complex changes and improve the quality of your models
Specification of Quality Assurance Techniques for SWM

Understand complex changes and improve the quality of your models
An example SWM model

Data Layer
- Address: street, postalCode, city
- Service: name, account, consultant
- BankAccount: number, bankCode, bankName
- Customer: name, email, address
- Motorbike: manufacturer, kind, cylinderCapacity
- Car: manufacturer, kind
- Employee: }
Potential Metrics for SWM

- **Context WebModel**
  - Number of Entities in the Model
  - Number of Dynamic Pages in the Model
  - Total number of Attributes in the Model
  - Total number of references in the Model
  - Average Number of Attributes per Entity
  - Average Number of References per Entity
  - ...

- **Context Entity**
  - Number of Attributes
  - Number of References
  - Number of Dynamic Pages referencing the Entity
  - ...

Understand complex changes and improve the quality of your models
Demo
EMF Refactor: New QA technique

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
New metric NEWM: Basic data

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
New metric NEWM: generated code

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
New metric NDPWM: Basic data

- **Target project**: `simplewebmodel.tutorial.metrics1`
- **Name**: NDPWM
- **Metric ID**: `models2013.tut.ndpwm`
- **Description**: Number of Dynamic Pages in the Web Model
- **Meta model**: `http://www.eclipse.org/simplewebmodel/1.0`
- **Context type**: `WebModel`

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
New metric NDPWM: OCL expression

```
OCL expression: DynamicPage.allInstances().size()
```
Potential Smells and Refactorings for SWM

- Empty Entity
- Unused Entity
- Entity without Data Page
- Entity without Index Page
- No Dynamic Page
- Missing Data Page
- Missing Index Page
- Equally Named Pages
- Multiple Link Definitions
- Missing Link
- ...

- Add Data Page to Index Page
- Add Index Page to Data Page
- Create Dynamic Pages
- Remove Multiple Links from Page
- Rename Page
- Update Links to Index Pages
- ...

Understand complex changes and improve the quality of your models
Henshin rule specification for SWM smell

No Dynamic Page

Henshin:
http://www.eclipse.org/henshin
Potential Smells and Refactorings for SWM

- Empty Entity
- Unused Entity
- Entity without Data Page
- Entity without Index Page
- No Dynamic Page
- Missing Data Page
- Missing Index Page
- Equally Named Pages
- Multiple Link Definitions
- Missing Link
- ...

- Add Data Page to Index Page
- Add Index Page to Data Page
- Create Dynamic Pages
- Remove Multiple Links from Page
- Rename Page
- Update Links to Index Pages
- ...

Understand complex changes and improve the quality of your models
Henshin rule specification for SWM refactoring Create Dynamic Pages

Rule mainUnit(selectedEObject, entitiname)

Understand complex changes and improve the quality of your models
Potential Smells and Refactorings for SWM

- Empty Entity
- Unused Entity
- Entity without Data Page
- Entity without Index Page
- No Dynamic Page
- Missing Data Page
- Missing Index Page
- Equally Named Pages
- Multiple Link Definitions
- Missing Link
- ...

- Add Data Page to Index Page
- Add Index Page to Data Page
- Create Dynamic Pages
- Remove Multiple Links from Page
- Rename Page
- Update Links to Index Pages
- ...

Understand complex changes and improve the quality of your models

Is the model complete?
An example SWM model

Data Layer

Address
  street
  postalCode
  city

Service
  name

BankAccount
  number
  bankCode
  bankName

Customer
  name
  email
  address
  account
  consultant

Employee

Motorbike
  manufacturer
  kind
  cylinder capacity

Car
  manufacturer
  kind

Hypertext Layer

CustomerData

EmployeeIndex

CarData

CarIndex

Starting page

Understand complex changes and improve the quality of your models
Improvements

- Refactoring: Add Data Page to Index Page on index page EmployeeIndex
- Refactoring: Add Index Page to Data Page on data page CustomerData
- Refactoring: Remove Multiple Links from Page on page StartingPage
- Refactoring: Update Links to Index Pages on page StartingPage
- Refactoring: Create Index Page and Data Page on entity Motorbike
- Manual Change: Remove entity Service
- Manual Change: Add attribute name to entity Employee
Understand complex changes and improve the quality of your models

Improved SWM model

Data Layer
- Address
  - street
  - postalCode
  - city
- BankAccount
  - number
  - bankCode
  - bankName
- Customer
  - name
  - email
  - address
  - account
  - consultant
- Motorbike
  - manufacturer
  - kind
  - cylinderCapacity
- Employee
  - name

Hypertext Layer
- EmployeeData
- CustomerData
- CarData
- CarIndex
- CustomerIndex
- Startingpage
- motorbikedata
- motorbikeindex

Is the model complete?
Demo
Tool support for understanding complex changes

**End-user tool support**

Model A

Model B

Matching → Difference Derivation → Recognition Rules → Edit Operation Detection → Edit Operations

**Meta-tools**: tool engineer support

Meta-model → Basic Edit Rules → Complex Edit Rules
Generated atomic edit rules

SERGe:
http://pi.informatik.uni-siegen.de/Mitarbeiter/mrindt/SERGe.php

Arendt, Kehrer, Taentzer
Understand complex changes and improve the quality of your models
Example complex edit rule

Understand complex changes and improve the quality of your models

Arendt, Kehrer, Taentzer
Generation of recognition rules

Understand complex changes and improve the quality of your models

Arendt, Kehrer, Taentzer
Generated recognition rules

Arendt, Kehrer, Taentzer

Understand complex changes and improve the quality of your models
Detected Edit Operations

Understand complex changes and improve the quality of your models

Arendt, Kehrer, Taentzer
Demo
Tool support for understanding complex changes

"Meta-tools": tool engineer support

Meta-model → Basic Edit Rules → Complex Edit Rules

End-user tool support

Model A → Matching → Difference Derivation → Low-level Changes → Edit Operation Detection → Edit Operations

Model B → Recognition Rules → Complex Edit Rules
Ecore-based representation of a difference

inv iv1: diff.modelA->contains(self.objA) and diff.modelB->contains(self.objB)

inv iv2: diff.modelB.contains(self.obj)

inv iv3: diff.modelA.contains(self.obj)

inv iv4: diff.modelB.contains(self.src) and diff.modelB.contains(self.tgt)

inv iv5: diff.modelA.contains(self.src) and diff.modelA.contains(self.tgt)

inv iv6: eContainer.correspondences->exists(c:Correspondence | c.objA = self.objA and c.objB = self.objB)
Representation of Semantic Change Sets
Edit Rule 2 Recognition Rule (Example)

Understand complex changes and improve the quality of your models
Understand complex changes and improve the quality of your models
Conclusion
Content of this tutorial

- Part I: Analyze and improve UML models
  - Quality assurance of UML models using
    - *model metrics, smells, and refactorings*
  - Understanding complex model changes by
    - *recognizing edit operations*

- Part II: Analyze and improve DSL models
  - Example DSL for simple Web applications
  - Specifying and applying new QA techniques
  - Specifying edit operations to understand model changes
Implemented model quality assurance techniques

<table>
<thead>
<tr>
<th></th>
<th>Ecore models</th>
<th>UML 2.3 models</th>
<th>UML 2.4 class models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics</td>
<td>23</td>
<td>106</td>
<td>23</td>
</tr>
<tr>
<td>Smells</td>
<td>3</td>
<td>30</td>
<td>27</td>
</tr>
<tr>
<td>Refactorings</td>
<td>22</td>
<td>30</td>
<td>27</td>
</tr>
</tbody>
</table>
Tool support for understanding complex changes

"Meta-tools": tool engineer support

End-user tool support

Understand complex changes and improve the quality of your models
Available edit operations to understand complex model changes

<table>
<thead>
<tr>
<th></th>
<th>UML 2.4 Class Diagrams</th>
<th>UML 2.4 State Machines</th>
<th>Ecore Models</th>
<th>SysML Block Definition Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Atomic</strong></td>
<td>259</td>
<td>264</td>
<td>148</td>
<td>637</td>
</tr>
<tr>
<td><strong>Complex</strong></td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td><strong>Refactorings</strong></td>
<td>3</td>
<td>1</td>
<td>32</td>
<td>-</td>
</tr>
</tbody>
</table>
Model quality assurance: outlook

- Specification and application of modeling guidelines
  - by complex edit operations

- Synchronized refactoring of model and code

Understand complex changes and improve the quality of your models
Understanding complex changes: outlook

High-level changes: edit operations
Low-level model changes
„Understand complex changes and improve the quality of your models!“

…to be able to work
- with large models
- in large teams

EMF Refactor:  
http://www.eclipse.org/emf-refactor

SiLift:  
http://pi.informatik.uni-siegen.de/Projekte/SiLift

Thorsten Arendt  
Timo Kehrer  
Gabriele Taentzer
Acknowledgement

This work is partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future - Managed Software Evolution.