A MODEL MANAGEMENT FRAMEWORK FOR MAINTAINING TRACEABILITY LINKS

Thomas Beyhl, Regina Hebig, Holger Giese

SE 2013
Workshop Traceability - Nutzung und Nutzen
STATE OF THE ART IN MDE

• Diverse modeling and model transformation languages that are combined within model transformation chains

• Diverse traceability approaches that work well for a certain combination of languages or model transformation technology

• Model transformation chains consist of automated and manual activities

• affect which traceability approaches can be used
CHALLENGES

• Automated maintenance of traceability links due to huge amount of development artifacts

• Combination of traceability approaches that work well for a certain combination of languages
  • runtime traceability approaches rarely supported by model transformation technologies
  • heuristic traceability approaches are imprecise

• Common format to combine traceability information for analysis of the whole set of artifacts
  • enable a generic processing of traceability links independently from (modeling) languages and model transformation technologies
REAPPEARING NEEDS

• How can traceability links be created and maintained as precondition for further research projects?

• Maintenance framework for traceability links that
  • can be reused in different research projects
  • can be configured individually to specific research needs
BASIC IDEA I
MEGAMODELS AS COMMON FORMAT

• Use megamodels as common format to store traceability links as defined by Barbero [1]

• Problem: A megamodel does not consider the content of development artifacts (too coarse grained)

cyf. [1]
BASIC IDEA I
HIERARCHICAL MEGAMODEL AS COMMON FORMAT

- Hierarchical megamodels take the content of development artifacts into account (more fine-grained)[4]
- Combination of megamodel, high-level and low-level traceability models
- Used as common format for storing traceability links
BASIC IDEA II
COMBINATION OF DIFFERENT SOURCES OF INFORMATION

• Sources for information
  • physical artifacts in the local workspace
  • model transformation results, e.g. trace model as by-product
  • logical information in the megamodel
BASIC IDEA II
COMBINATION OF DIFFERENT SOURCES OF INFORMATION

• Observation of artifacts
  • creation, deletion and modifications of artifacts

• Observation of model transformations
  • translation of traceability information into common format

• Analysis of megamodel using traceability approaches
  • heuristic analysis on logical level
ARCHITECTURE

Execution Layer

- Standard Traceability Adapter
- Traceability Adapter
- Heuristic Traceability Adapter

Logical Representation

- Megamodel
- Maintenance Interface
- Synchronisation Interface
- Traceability Interface

Representation (Logical) Layer

- Standard Synchronisation Adapter
- EMF Model Synchronisation Adapter
- Synchronisation Adapter

Physical Layer

- Eclipse Workspace
- Monitoring Interface
- Modification Interface
- Synchronisation Interface
- Execution Interface

Execution Layer

- Xpand Adapter
- StoryDiagram Adapter
- TGG Adapter
- ATL Adapter

Maintenance Interface

Standard Traceability Adapter

Model Operation Adapter

Heuristic Traceability Adapter

EMF Model Synchronisation Adapter

Logical Representation

ATL Adapter
METAMODEL OF THE MEGAMODEL
EXAMPLE

Block diagram „ClientServerArchitecture“

Client connectsTo Server
EXAMPLE

(ARTIFACT TYPES AND ARTIFACT INSTANCES)
EXAMPLE

Block diagram „ClientServerArchitecture“

UML class diagram „ClientServerArchitecture“
EXAMPLE
(RELATION TYPES AND RELATION INSTANCES)

: MOM

relationTypes

: MOMRelationType
name = SDL2UML

artifactInstances

: MOMRelation
name = SDL2UML

: MOMWorkspaceProject
uri = /DemoProject/

: MOMRelation
name = SDL2UML

: MOMEMFModel
uri = /DemoProject/model.classdiag
name = ClientServerArchitecture

: MOMEMFModel
uri = /DemoProject/model.blockdiag
name = ClientServerArchitecture

: MOMMetaModel
uri = http://.../blockdiagram/1.0

: MOMMetaModel
uri = http://.../classdiagram/1.0

: MOMEMFModel
uri = /DemoProject/model.classdiag
name = ClientServerArchitecture

: MOMEMFModel
uri = /DemoProject/model.blockdiag
name = ClientServerArchitecture

: MOMParameter
name = input

: MOMParameter
name = output

: MOMParameter
name = input

: MOMParameter
name = output

value

source
target

instanceOf
MAINTENANCE OF THE MEGAMODEL

• Workspace / Repository throws notification when an artifact is created, deleted or modified

• Megamodel builder (incrementally) synchronizes the megamodel with the help of the artifact adapters to interpret the artifact’s content

• Megamodel is logical view on the physical workspace
TRACEABILITY LINK CREATION

• By model operation execution, e.g.
  • Extracting traceability information created as by product into
    common format (e.g. in case of ATL [2])
  • Translating correspondence graph into common format (e.g. in case of TGG)

• By change notifications of the workspace / repository

• Search for traceability links heuristically
• Traceability adapter which creates the traceability link is responsible for its maintenance

• Search for invalid relations within the hierarchical megamodel

• Missing value for source or target parameters after megamodel was synchronized with workspace

• Search on logical level (megamodel) vs. physical level (raw artifacts)
CONCLUSION

• Decoupling of languages and traceability approaches
  • arbitrary combination of traceability approaches

• Depending on the used technology different amount of trace information can be extracted

• Combination of traceability links enables the analysis for the whole set of artifacts

• Framework provides traceability information for other technologies, e.g. Composition of True-Black Box Model Operations [3], VCS, Build-Server [5]
REFERENCES


