Exposé for B.Sc. Thesis
Megamodel-driven Traceability Recovery & Exploration of Correspondence & Conformance Links

Maximilian Meffert
210101205

University of Koblenz-Landau

1 Introduction

This exposé[1] outlines the thesis:

Megamodel-driven Traceability Recovery & Exploration
of Correspondence & Conformance Links

for acquiring the degree Bachelor Science (B.Sc.) in Computer Science. The central topic of the thesis will be the study of traceability recovery in a megamodel governed environment. Furthermore a system for exploring the recovered traceability links will be developed.

1.1 Road-map

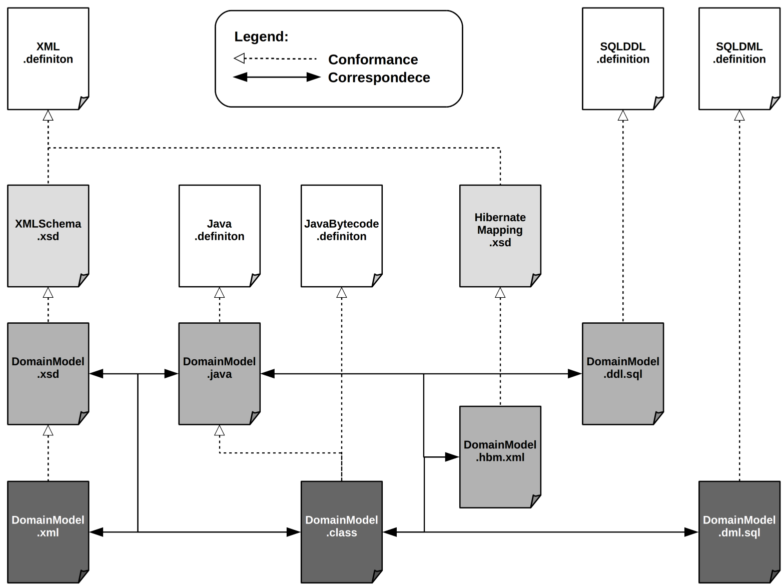
Section 2 motivates the topic of the thesis. 3 gives a short overview over the necessary background information. 4 defines a preliminary hypothesis and formulates the research questions. 5 specifies the important objectives for the thesis. 6 describes the approach and methodology of the thesis. 7 outlines the interim structure of the thesis.

2 Motivation

A common task during the development of software systems is to persist and serialize a domain model. For instance, consider a simple ReST-ful web-service where data is stored in a database and served via HTTP in serialized form, e.g. XML. Given such a system, one can observe correspondences (structural similarities) and conformances (compliance with a definition) between different artifacts, i.e. manifestations of the same domain model. Figure 1 shows a non exhaustive depiction of such relationships in O/R/X scenario.

Upon closer inspection, one can also observe that correspondence and conformance are not just interrelations of the artifacts at large. In fact the same relations can be found between fragments of the artifacts. Table 1 shows fragments corresponding to a Java class property in the O/R/X scenario.

[1] `http://www.softlang.org/info:expose`



The depicted relationships may not be exhaustive.

Fig. 1. O/R/X Correspondence & Conformance

Language	**Fragment Type**	**Fragment Text**
Java	Class Property	public String name;
XSD	Attribute Definition	<xs:attribute name="name" type="xs:string"/>
XML	Attribute	name="Alan Turing"
Hibernate XML	Property Mapping	<property access="field" generated="never" lazy="false" name="name" optimistic-lock="true" type="java.lang.String" unique="false"> <column name="name"/> </property>
SQL/DDL	Column Definition	‘name‘ varchar(255) DEFAULT NULL,
SQL/DML	Column Definition	INSERT INTO Employee (...,name,...) VALUES (...,"Alan Turing",...)

Table 1. Corresponding Fragments in an O/R/X Scenario

The motivating idea behind the thesis is that correspondence and conformance relationships between artifacts create trace-links left behind by transformations in the sense of [4]. Thus, it should be possible to recover these links with a model describing such relations. A technology capable of modeling such *"linguistic architectures"*[2] is MegaL[2] [5].

3 Background

The thesis will be based but is not limited to aspects of the following topics:

[2] http://www.softlang.org/megal/

- **Traceability** The ability to interrelate artifacts of a software development process [1][8]. The automatic process of relating artifacts is called traceability recovery and recovered elements of the relationships are called links. Traceability support is one desired capability of MegaL and serves as main subject of the thesis.
- **Megamodeling** The process of interrelating models, e.g. relationships between models, meta-models, meta-meta-models up to meta*-models. Megamodeling in the sense of MegaL mainly interrelates software languages and kindred objects [2][5].
- **Ontologies** The systematic modeling and representation of knowledge [3]. Megamodels as used in the thesis are in fact some sorts of software linguistic ontologies, so ontologies may be covered as related work. Ontologies also utilize mereology.
- **Mereology** The study of logic part-whole relationships [7]. Merology serves as theoretical basis for modeling fragments of linguistic artifacts.
- **Program Analysis** The process of automatically analyzing programs either statically, by analyzing source code, or dynamically by monitoring and intercepting the runtime of a program, i.e. analyzing transient artifacts. Traceability recovery of linguistic links is in fact an application of program analysis.
- **XML Data Binding** The process of automatically mapping model- and instance-level objects into a XML-serialized form. XML Data Binding as implemented by JAXB[3] will serve as a scenario subjected to traceability recovery of linguistic links.
- **Object Relational Mapping** The process of automatically mapping model- and instance-level objects into a relational data-storage. Object Relational Mapping as implemented by JPA[4] and Hibernate[5] will serve as another scenario subjected to traceability recovery of linguistic links

4 Research Hypotheses & Questions

Parthood is the relationship between a whole and its constituent parts. [7] and [4] axiomatize it as a reflexive, antisymmetric and transitive relation:

$$
\begin{aligned}
&x \ \mathsf{partOf}\ x \\
&x \ \mathsf{partOf}\ y \wedge y \ \mathsf{partOf}\ x \Rightarrow x = y \\
&x \ \mathsf{partOf}\ y \wedge y \ \mathsf{partOf}\ z \Rightarrow x \ \mathsf{partOf}\ z
\end{aligned}
$$

the A stricter, irreflexive form of parthood is the proper-part relationship [7]:

$$x \ \mathsf{properPartOf}\ y \Leftrightarrow x \ \mathsf{partOf}\ y \wedge \neg(y \ \mathsf{partOf}\ x)$$

[3] Java Architecture for XML Binding `https://docs.oracle.com/javase/tutorial/jaxb/intro/arch.html`

[4] Java Persistence API `http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html`

[5] `http://hibernate.org/`

Since software artifacts are based on formal languages it should also be fair to assume atomicity [7]:

$$\forall x \exists y : y \ \mathsf{partOf} \ x \land \neg(\exists z : z \ \mathsf{properPartOf} \ y)$$

Objects with no proper parts are called atoms, otherwise their called fusions.

Correspondence is the relationship between two software artifacts who share a structural similarity. [4] axiomatizes it as a strict one-to-one relation between software artifacts:

$$\begin{aligned}
&(a_1, a_2) \in R \subseteq L_1 \times L_2 \\
&\land \forall b_1 \in L_1 : b_1 \ \mathsf{partOf} \ a_1 \Rightarrow (\exists! b_2 \in L_2 : b_2 \ \mathsf{partOf} \ a_2 \land b_1 \ \mathsf{correspondsTo}_R \ b_2) \\
&\land \forall b_2 \in L_2 : b_2 \ \mathsf{partOf} \ a_2 \Rightarrow (\exists! b_1 \in L_1 : b_1 \ \mathsf{partOf} \ a_2 \land b_2 \ \mathsf{correspondsTo}_R \ b_1) \\
&\Rightarrow a_1 \ \mathsf{correspondsTo}_R \ a_2
\end{aligned}$$

Given an arbitrary relationship R between to languages, for instance a transformation from one language to the other, two artifacts correspond to each other if and only if for each part of one artifact there exists exactly one corresponding part of the other.

Conformance is the relationship between two software artifacts, denoting that one artifact complies to the definition given by the other. [4] axiomatizes it as follows:

$$\forall x \in \mathsf{Any} : x \in L \subseteq \mathsf{Any} \Leftrightarrow \exists d \in D \subseteq \mathsf{Any} : x \ \mathsf{conformsTo} \ d$$

Given a set Any serving as a universe for software artifacts, an arbitrary artifact conforms to another one if and only if the latter artifact is a definition of a software language whom the former is an element of.

One objective of the thesis should be to provide empirical assurance for the axiom above in the sense that correspondence and conformance are in fact to some extend mereologically induced. However, [4] also notes that strict correspondence may be unrealistic since real world artifacts may contain two or more parts corresponding to only one part in another artifact, e.g. an XSD documents can contain an element and a complex type definition corresponding to only on Java class declaration. For this reason the thesis will assume a weaker forms of correspondence and conformance as research hypotheses:

RH1 **Fragment Correspondence Hypothesis**

$$\begin{aligned}
&\forall a_1 \in L_1, a_2 \in L_2 \exists b_1 \in L_1, b_2 \in L_2 : \\
&a_1 \ \mathsf{correspondsTo}_R \ a_2 \Rightarrow b_1 \ \mathsf{partOf} \ a_2 \land b_2 \ \mathsf{partOf} \ a_2 \land b_1 \ \mathsf{correspondsTo}_R \ b_2
\end{aligned}$$

If two artifacts are assumed to correspond to each other, then parts of both artifacts should exist which also correspond to each other.

RH2 **Fragment Conformance Hypothesis**

$$\begin{aligned}
&\forall a_1 \in L, a_2 \in D \exists b_1 \in L, b_2 \in D : \\
&a_1 \ \mathsf{conformsTo} \ a_2 \Rightarrow b_1 \ \mathsf{partOf} \ a_2 \land b_2 \ \mathsf{partOf} \ a_2 \land b_1 \ \mathsf{conformsTo} \ b_2
\end{aligned}$$

> If two artifacts are assumed to be in a conformance relationship, then parts of both artifacts should exist which share the same relationship.

This leads to the following research questions:

RQ1 Is correspondence to some extend strictly mereologically induced?

$$\forall a_1 \in L_1, a_2 \in L_2 \exists b_1 \in L_1, b_2 \in L_2 :$$
$$a_1 \ \mathsf{correspondsTo}_R \ a_2$$
$$\Rightarrow b_1 \ \mathsf{properPartOf} \ a_2 \wedge b_2 \ \mathsf{properPartOf} \ a_2 \wedge b_1 \ \mathsf{correspondsTo}_R \ b_2$$

RQ2 Is conformance to some extend strictly mereologically induced?

$$\forall a_1 \in L_1, a_2 \in L_2 \exists b_1 \in L_1, b_2 \in L_2 :$$
$$a_1 \ \mathsf{conformsTo} \ a_2$$
$$\Rightarrow b_1 \ \mathsf{properPartOf} \ a_2 \wedge b_2 \ \mathsf{properPartOf} \ a_2 \wedge b_1 \ \mathsf{conformsTo} \ b_2$$

5 Objectives

Objectives for the thesis are:

O1 Implementation of a MegaL/Xtext-extension capable of recovering traceability links representing parthood, correspondence and conformance relationships between code fragments.
O2 Implementation of a MegaL/Xtext-extension allowing for an user to visually explore traceability links, i.e. parthood, correspondence and conformance relationships between code fragments
O3 Providing an extensive discussion comparing MegaL with related approaches on traceability recovery.
O4 Providing an extensive discussion comparing MegaL with related approaches on ontologies for software artifacts or software engineering in general.
O5 Providing an answer for the research questions.

6 Methodology

Thesis and research will utilize an example-driven approach inspired by the 101system[6]. For this, the system to study will be an imaginary Human Resource Management System (HRMS). Figure 2 shows an UML-Class-Diagram depicting the model of this system.

The HRMS will be implemented in plain Java with two scenarios in mind: (a) XML-Binding with JAXB and (b) Persistence with JPA/Hibernate. Both scenarios will be studied with a concrete instance of the HRMS model.

[6] `https://101wiki.softlang.org/101system`

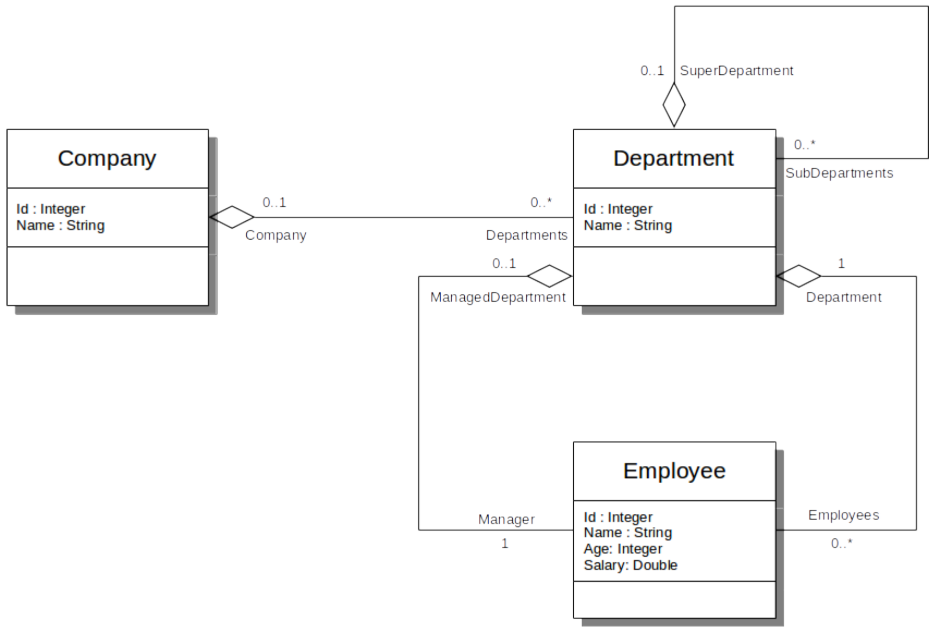


Fig. 2. The Human Resource Management System Model

7 Structure of the Thesis

The interim structure[7] of the thesis is outlined by but not limited to the chapters as follows:

1. **Introduction** This chapter will introduce the thesis subject, also delimiting possible contributions to traceability recovery and megamodeling.
2. **Background** This chapter will provide the theoretical background of the thesis as outlined but not limited to the topics in section 3 above.
3. **Related Work** This chapter will provide a sound discussion of related work regarding megamodeling and traceability recovery of linguistic relationships between software artifacts.
4. **Methodology** This chapter will describe the approach of the thesis as outlined in section 6 above.
5. **Requirements** This chapter will define the requirements and acceptance criteria for the system to be developed as described in section 5.
6. **Design** This chapter will provide a description of the design and architecture for the developed system from section 5.
7. **Implementation** This chapter will provide explanations of non trivial implementation details of the developed system, e.g. a fragmentation algorithm for software artifacts could be necessary for the system.

[7] `http://softlang.wikidot.com/info:thesis-structure`

8. **Results** This chapter will provide a discussion of empirical results gathered by the implemented system from section 5 applied to the scenarios described in section 6.
9. **Conclusion** This chapter will summarize the thesis, pointing out limitations and possible future work.

References

1. Ieee standard glossary of software engineering terminology. IEEE Std 610.12-1990 pp. 1–84 (Dec 1990), http://ieeexplore.ieee.org/servlet/opac?punumber=2238
2. Favre, J., Lämmel, R., Varanovich, A.: Modeling the linguistic architecture of software products. In: Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings. pp. 151–167 (2012), http://dx.doi.org/10.1007/978-3-642-33666-9_11
3. Guarino, N., Oberle, D., Staab, S.: What is an ontology? In: Handbook on Ontologies, pp. 1–17 (2009), https://doi.org/10.1007/978-3-540-92673-3_0
4. Lämmel, R.: Coupled software transformations revisited. In: Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering, Amsterdam, The Netherlands, October 31 - November 1, 2016. pp. 239–252 (2016), http://dl.acm.org/citation.cfm?id=2997366
5. Lämmel, R., Varanovich, A.: Interpretation of linguistic architecture. In: Modelling Foundations and Applications - 10th European Conference, ECMFA 2014, Held as Part of STAF 2014, York, UK, July 21-25, 2014. Proceedings. pp. 67–82 (2014), http://dx.doi.org/10.1007/978-3-319-09195-2_5
6. Rector, A., Welty, C., Noy, N., Wallace, E.: Simple part-whole relations in owl ontologies (August 2005), http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/simple-part-whole-relations-v1.5.html, retrieved 12. June 2017
7. Varzi, A.C.: Parts, wholes, and part-whole relations: The prospects of mereotopology. Data Knowl. Eng. 20(3), 259–286 (1996), http://dx.doi.org/10.1016/S0169-023X(96)00017-1
8. Winkler, S., Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. Softw. Syst. Model. 9(4), 529–565 (Sep 2010), http://dx.doi.org/10.1007/s10270-009-0145-0