Using Metamodels Formal Semantics for representing Rationale during the Software Design

Adriana Pereira de Medeiros, Daniel Schwabe
TecWeb Laboratory, Department of Informatics, PUC-Rio
Rio de Janeiro, RJ – Brasil
{adri,dschwabe}@inf.puc-rio.br

Abstract. This work proposes the use of the metamodels formal semantics in the representation of design rationale during model-based designs, particularly software design. This is achieved through the integration of the design rationale representation model described by the Kuaba ontology with the artifact’s formal semantics, as provided by the metamodel of the design methods (or modeling language) used. This integration makes the design rationale representations more specific according to the design methods and enables the computational processing of the recorded rationale to support the design of new artifacts. Moreover, the use of the metamodels formal semantics can reduce the overheads of DR authorship, since a greater part of the rationale can be automatically obtained from the meta-model used.

1. Introduction

Design Rationale (DR) represents the reasons behind design decisions. The research developed in the DR area seeks to provide models and tools that allow explicitly recording these reasons, in order to support its use in the design of new artifacts.

In Software Engineering, DR represents a key point for the software development understanding, since it facilitates its maintenance, evolution and reuse in new projects. DR makes the decision elements explicit and, thus, facilitates the collaboration between the developers. It allows analyzing the possible solutions for the design of a particular artifact, pointing the advantages and disadvantages of each solution alternative. Moreover, the DR capture permits the developers to explicate their knowledge and later examine the justifications of some decision. This is important for many reasons, including the fact that, in many cases, over a longer period of time, the designers themselves may not recall all the rationale they used in the design of a particular artifact.

Although there are several proposals in the literature for capturing and representing DR, such as IBIS [Kunz and Rittel 1970], DRL [Lee 1991], RatSpeack [Burge 2004] and TEAM [Lacaze 2005], most of them generate incomplete and informal representations, not enabling computational processing over the represented rationale. Furthermore, when applying them to formally defined artifacts (such as software), their informality prevents automatically taking into consideration alternatives prescribed by the design methods, as well as incorporating their restrictions. In other words, it is not possible to leverage the semantics of the artifact provided by the formal model that describes it.
This work presents the Kuaba approach [Medeiros 2006], a new approach for representing DR that integrates the DR representation model with the formal semantics provided by the metamodel of the design method or modeling language used for describing the artifact being designed. This approach seeks to reduce the effort required from the designer during the capture and representation of DR and to permit the computational processing of captured rationale for supporting model-based designs, particularly, software design.

Model-based design is a special kind of design domain in which design can be seen as an instantiation process of a metamodel. This metamodel represents the formal models that describe the designed artifacts and provide semantic descriptions which allow reasoning over the artifacts being produced. An example of such a formal model in software design is the metamodel of the Unified Model Language (UML).

Considering the design as an instantiation process of a metamodel, we can integrate the formal semantics of this metamodel with the design process to provide a more automated support for the designers, so that they can record their rationale while designing the artifacts, using a single tool. Moreover, the use of the metamodel formal semantics in the DR representation enables a new type of design reuse at the highest abstraction level, where rationales are processed and re-employed to support the design of new artifacts.

In the remainder of this paper, we first present the Kuaba approach and the DR representation model used. Next, we address the issue of how the formal semantics of the metamodel that describes the artifacts can be integrated with the representation model for recording DR. Then, we briefly discuss the architecture of an integrated environment for supporting the DR processing. Finally, we conclude by discussing related work and drawing some conclusions pointing out further work.

2. The Kuaba Approach

Kuaba is an argumentation-based approach for representing DR. The main objective of this approach is to permit the representation and the use of DR to support model-based designs, particularly, software design.

The Kuaba approach integrates the DR representation model defined by the Kuaba ontology [Medeiros et al. 2005] with the formal semantics of the artifacts provided by the metamodel of the design method or modeling language that describe them. This integration is performed by the use of this formal semantics in the instantiation of the Kuaba ontology that is described in F-Logic [Kifer and Lausen, 1989], a formal language for ontology specification.

Theoretically, when formal semantics for the artifacts are available, fully automated systems could be constructed to automatically synthesize artifacts, but this is not always feasible.
neither the approach nor the focus taken in this paper. Kuaba explicitly requires human intervention in defining design steps or operations in producing the final design.

2.1 The Kuaba Ontology

The Kuaba ontology formally specifies a knowledge representation model for supporting the recording of DR in model-based designs. It is composed by a vocabulary and a set of rules described in F-logic. The elements of the vocabulary are illustrated in Figure 1 using an UML-like graphical notation to help visualization. Notice that such object oriented model is used only as a suggestion for presenting the ontology vocabulary; some relations and constraints were hidden to simplify its presentation.

In the Kuaba ontology vocabulary, the reasoning elements represent the design questions discussed during the artifact production, the possible solution ideas that address these questions and the arguments against or in favor of the presented ideas. The decision element records if a particular idea was accepted or rejected as a solution for a question, and must be associated to a final justification that is always based on the arguments presented for the idea. The artifact element represents a final design solution, made up by the set of accepted ideas in the DR representation. Artifacts are represented by two elements, Atomic Artifact and Composite Artifact. These elements are instantiated according to formal semantics of the artifact being produced. For example, in the UML metamodel a class can be seen as an aggregate of attributes and, therefore, an information item modeled as a class can be represented as a composite artifact. The method element describes the method or process of design used in an application modeling, for instance, the Unified Software Development Process [Jacobson et al. 1999]. Finally, the formal model element represents the metamodel of the design method or modeling language used for describing the artifact.

Below we show a portion of the Kuaba ontology shown in Figure 1 expressed using F-Logic.
Notice that each element has a set of properties and relations that compose the structure of the rationale developed by the designer during the design. For example, the element Question has a “hasType” property, with possible values “AND”, “OR” and “XOR”. The value “XOR” indicates that all ideas that address this question (i.e., are possible answers) are mutually exclusive, meaning that only one idea can be accepted as a solution to the question. The value “AND” indicates that the designer should accept all ideas that address the question or reject all of them. Finally, the value “OR” indicates that various ideas can be accepted as a solution to the question. This kind of information allows us to define rules that can suggest decisions about the acceptance or not of the proposed solution ideas, such as the second rule (axiom) shown above. This rule is defined to support decisions for questions of the type “XOR”. According to this rule, if an idea associated with a question of type “XOR” is accepted by the designer, then all other ideas associated with this question must be rejected. This rule is used for supporting decisions about the acceptance or not of the solution ideas proposed for a particular question during the design.
3. Using Metamodels Formal Semantics to represent Rationale

In this section, the use of the Kuaba approach to the design rationale representation is illustrated considering the conceptual model design of a Web application for the recording and searching of movies and TV series, called IMDB (Internet Movie Database). The application conceptual model is represented as an UML class diagram, as Figure 2 shows.

![Figure 2 – Partial Class Diagram modeling the IMDB application](image)

During the design of this artifact, the designers identify the domain information items that are relevant for the design, which represent the possible elements of the conceptual model being created. These information items are determined by the designers’ knowledge of the domain. They could also be obtained from domain ontologies, or extracted from the DR of a previous phase in the software process, requirements elicitation, which is not addressed in this paper. In this example, we consider that the designers initially propose the items Actor, Feature, Movie and Series. Next, the designers must decide how each one of them will be modeled using the UML to make up the final artifact, the class diagram.

Since the artifact being designed is represented as a class diagram, the formal semantics of the UML metamodel is used in the Kuaba ontology instantiation for describing the rationale behind this artifact design. Figure 3 shows part of the UML metamodel that is used in the DR representation example presented in this section (see [OMG 2003] for the complete metamodel).
Figure 3 – Partial UML metamodel for class diagrams

According to this part of the UML metamodel an element in a class diagram can be a “Class”, an “Attribute”, an “Association”, a “Generalization” or an “Association Class”. By the inheritance relation, an element of the type “Class” can be seen as an aggregate of elements of the type “Attribute”. Similarly, an element of the type “Association” can be seen as an aggregate of elements of the type “Association End”, where each “Association End” must specify the class element that participates of the association. These definitions represent design options that can be used to model the information items of the knowledge domain in which the designers are working. These design options are used in the Kuaba ontology instantiation for representing the DR for the conceptual schema being designed.

The DR representation typically starts with a general question that establishes the problem to be solved. According to the UML metamodel (Figure 3), the first problem to be solved in designing a class diagram is the identification of its constituting elements. Applying the Kuaba ontology vocabulary, this results in instantiating the “Question” element with the instance “What are the model elements?”. Figure 4 depicts a graphical representation we have created to help visualizing instances of the Kuaba ontology, showing the portion of the DR regarding the solution ideas to model the “Actor” information item. In this representation, the root node is an initial question (represented as rectangles), “What are the model elements?”, which is addressed by the ideas “Actor”, “Photo”, “Name” and “Biography”, represented as ellipses.

Once these first ideas for the conceptual model elements have been established, the designers must decide how each one of them will be modeled using the UML. This next step is represented in Figure 4 by the “suggests” relation, which determines questions of the type “How to model ...? ” entailed by proposed ideas.
Figure 4 – Example of Design Rationale for the Actor element

The possible ideas that address these questions are determined by the UML metamodel for class diagrams – elements can essentially be a class, an attribute, or an association. Accordingly, the “Class” and “Attribute” ideas linked to the questions “How to model ...?” are established as an instantiation of the “Idea” element. Since the “Attribute” idea, in turn, must be associated with a “Class” according to the UML metamodel, the question “Whose?” is suggested, which in turn will be addressed by the idea corresponding to the class whose attribute it is. For instance, the information item “Biography” can be modeled as an attribute of the item “Actor” modeled as a class. Similarly, the questions “Minimum Multiplicity?” and “Maximum Multiplicity?” are also defined by the UML metamodel to be associated with the idea “Attribute”. The arguments for and against each option (represented as dashed rectangles in Figure 4) record the experiences and the knowledge employed by the designers in the artifact design.

Figure 4 also shows the decisions made, labeling each solution idea to each question with an “A” (for accepted) or “R” (for rejected). Thus, the example represents the fact that the designers decided to accept the “Attribute” idea as a solution to the question “How to model Photo?”, in detriment of the “Class” idea.

The portion of the Kuaba ontology instance shown below represents part of this rationale expressed using F-Logic. Observe that the reasoning elements based on the UML metamodel have the “isDefinedBy” property that distinguishes them from the reasoning elements provided by the designers. This property is used in the rules and operations defined to support the DR processing.
Figure 5 shows part of the rationale used for modeling the information item “Feature”. Analyzing this rationale we can observe that the designers decided modeling “Title” and “Description” as attributes of a class “Feature”, including a generalization relation between this class and the classes “Movie” and “Series”.

Figure 5 - Example of Design Rationale for the Feature element
According to the UML metamodel (Figure 3), a generalization is a relation between an element “parent” and at least one element “child” that inherits the properties defined in the “parent” element. Generally, these elements are known in object oriented modeling as “superclass” and “subclass”, respectively. However, we use the terms “parent” and “child” adopted in the UML metamodel as instance values for the “Question” element of the Kuaba ontology, since the generalization concept also can be used to represent specializations of associations between classes. Thus, the DR represented in Figure 5 shows that the designers accepted the solution of modeling the “Feature Type” element as a generalization relation between the element “Feature” (parent) and the elements “Movie” and “Series” (children).

Figure 6 shows the rationale behind the decisions about the design of the information item “Role”. In this example, we can see that the designer considered and rejected the idea of modeling an association “Acts on” between the elements “Actor” and “Feature”. In addition, he considered the idea of including an element “Role” in his conceptual model and the ideas “Attribute”, “Class” and “Association Class” as possible design options for modeling this item in the class diagram. According to the UML metamodel, shown in Figure 3, an association class is a specialization of the elements “Class” and “Association”. As a specialization of the “Association” element, an association class must have at least two association ends. Thus, the type attribute of the question “Association Ends?” shown in Figure 6 must be recorded in the rationale representation with value “OR”, since more than one solution idea must be accepted for this question. In this example, the decisions of accepting or rejecting the ideas “Destination” and “Origin” must be the same, i.e., they must be both accepted or both rejected.

Figure 6 – Example of Design Rationale for the Role element
In the UML metamodel, each end proposed for an association must be related to one of the classes participating of the association. This is represented by the questions “Participant?” shown in Figure 6. The ideas that address these questions indicate that the designers considered modeling the “Role” element as an association class between the “Actor” and “Feature” elements, both modeled as classes. However, this solution idea was rejected according to the argument presented against the idea “Association Class”. Figure 6 shows the rationale behind other decisions made about the design of the “Role” element, such as the decision of accepting the idea “Class” to model this element in the class diagram.

These examples show that when there is a well defined metamodel to describe an artifact, we can take advantage of its semantics to represent the DR. This allows the creation of a semi-automated process, in which a support environment uses the formal semantics of the metamodel to suggest design options at each step in the design, and records the corresponding choices made by the designer using the vocabulary pre-defined by the Kuaba ontology. This process can also facilitate the DR capture, since it allows automating part of the generation of DR representations. Thus, the large amount of data produced in DR representations of actual designs is significantly hidden from the designer through the use of automated support.

4. Supporting the Design Rationale Representations Processing

The DR processing is supported by a set of computable operations, implemented according to the vocabulary and formal rules of the Kuaba ontology. These operations are performed by the rationale processor, one of the components of the conceptual architecture of the integrated design environment proposed in [Medeiros 2006]. The goal of this environment is making the capture, representation and use of DR part of the design process and, consequently, to reduce the effort required from the designer to record his reasoning. Figure 7 shows the conceptual architecture of this integrated environment.

![Figure 7 – Conceptual Architecture of an Environment supporting DR](image)

This architecture proposes the extension of the existing software design tools to support DR, since most of them already use some kind of formal description (metamodel) of the artifacts being designed. This extension enriches the design tools by adding two layers to support the editing and searching of DR. In the editing layer, the
designer informs the arguments for and against the design alternatives considered, and
the justifications for the decisions made. Notice that the designer does not need to
inform the questions and design ideas considered, since they are automatically captured
from the metamodel by the design tool while he creates his artifact. In the search layer,
he searches existing designs with their rationales, formulates questions about the
designs found, and starts the integration of rationales to produce a new artifact.

In the proposed architecture, the design tool transfers the design options and the
rationale information provided by the designer to the rationale processor, responsible
for creating the DR representations and processing the rationales integration, when
requested by the designer. The integration of DRs involves matching two or more DR
representations for designing a new artifact.

5. Related Work

The integration of the DR representation with semantics provided by the software
design methods was also proposed in the Potts and Bruns model [Potts and Bruns 1988],
which was extended by [Lee 1991] in defining DRL. However, the Potts and Bruns
model and the Kuaba approach differ in the way they use this semantics. In Potts and
Bruns, the generic model entities are refined to accommodate a particular design
method’s vocabulary for deriving new artifacts. For example, a new entity specific to
the design method used is incorporated into the IBIS model. In the Kuaba approach the
formal semantics of the metamodel prescribed by the design method is used in the
instantiation of the reasoning elements (Question and Idea), which allows to automate
the generation of part of the rationale that is informed by designers during the artifact
design. In other words, Kuaba works at the metamodel level, whereas the Potts and
Bruns approach works at the model level, requiring a change in the vocabulary for each
different software design method.

6. Conclusions

In this paper we have presented a new approach for the DR representation, in which DR
is represented in a formal specification language incorporating the formal semantics
provided by the metamodel of the design method or modeling language used for
describing the artifact being designed.

The approach Kuaba provides a meaningful gain in the expressive power of the
DR representations. The use of the metamodel formal semantics makes the DR
representations more specific according to the software design methods and permits the
processing of DR to support the design of new artifacts. It can also facilitate the DR
capture, since it allows automating part of generation of DR representations. Therefore,
the large amount of data produced in DR representations of actual designs is
significantly hidden from the designer through the use of automated support. This
automated support can reduce the overheads of DR authorship, since a greater part of
the rationale (questions and design ideas) is automatically obtained from the meta-
model used by the design tool.

Our further work includes: the investigation of the use of the Kuaba approach to
represent DR considering other activities of the software development process, the
accomplishment of an empirical validation of the approach and the investigation of the
use of Kuaba in other domains, with different kinds of meta-models, such as
engineering and geophysics. The goal here is to further experiment with the semantics of the meta-models and the degree of automation it enables within the support environment.

**References**


