A Design Rationale Representation for Model-Based Designs in Software Engineering

Adriana Pereira de Medeiros, Daniel Schwabe, and Bruno Feijó

Dept. of Informatics, PUC-Rio, Rua Marquês de São Vicente 225,
22453-900, Rio de Janeiro - RJ, Brasil
{adri, dschwabe, bruno}@inf.puc-rio.br

Abstract. This paper presents the Kuaba Ontology, a vocabulary for Design Rationale described in an ontology definition language that allows attributing semantics to recorded content, and defining rules that enable performing computable operations and inferences on this content. This vocabulary extends the argumentation structure of the Issue Based Information System (IBIS) explicating the representation of the decisions made during design and their justifications, and the relations between the argumentation and generated artifacts. Furthermore, we propose to support the design process through the use of the semantic descriptions defined by formal models of the artifacts. Representing Design Rationale using an ontology definition language and the artifacts formal model, enables a type of software reuse at the highest abstraction level, where rationales are re-employed in designing a new artifact. This kind of reuse is possible in knowledge domains where there are formal models describing the artifacts, in particular, in the Software Design domain.

1 Introduction

Designing a software artifact typically involves understanding the problem being addressed, identifying possible solution alternatives, analyzing them, and deciding which solutions will be used to construct the final artifact. The final products of this process, the artifact and its specifications, represent the final solution chosen for the particular design problem, but do not represent the reasons that led the designers to choose that one among the other available alternatives, and why the others were discarded. In other words, they do not capture the Design Rationale (DR).

DR is the reasons behind design decisions. In most cases the DR is not adequately documented, which leads to requiring a high degree of verbal communication among persons that must work with an artifact, in order to understand the reasoning followed by the designer. For instance, this is fundamental when maintaining a software artifact designed by another person, or when trying to reuse it in the context of a new design. Therefore, recording the DR during the design is critical to allow its reuse.

There are several proposals in the literature for representing DR, such as IBIS [1] and DRL [2]. Most of them are incomplete or informal, not enabling machine-processable computations over the represented DR. Consequently, it is not possible to guarantee that the representation is consistent and even that it does actually provide some sort of explanation about the captured design. Furthermore, when applying them

Proceedings of the CAiSE'05 Forum - O. Belo, J. Eder, J. Falcão e Cunha, O. Pastor (Eds.)

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.

For many knowledge domains, particularly in software design, there are formal models that describe the artifacts and present semantic descriptions, which allow reasoning over the artifacts being produced. In this paper, this special type of design domain is called "model-based design". An example of such a formal model is the UML specification language [3] used to describe a class diagram. A formally defined DR representation may allow integrating the formal semantics of the artifacts being designed, and allows automated computations over such representations. When such representations are available in a distributed environment, it is possible to envisage the collaboration between designers with semi-automated support, where DR representations can be searched for, recovered and integrated during the process of designing a new artifact. The integration of different DRs is possible only if the following conditions are satisfied: the artifacts are build from the same type of formal model (e.g. two UML class diagrams); the DRs are used to represent the same domain of application (e.g. a CD catalogue) and the DR of the artifacts is represented using the same (or compatible) representation scheme(s) - e.g. an ontology vocabulary.

A semi-automated computational environment is being built to support designers through processing of formal DR representations. This environment uses the formal model for the artifact being designed to suggest design options at each step in the design, and records the corresponding choices made by the designer, using a special purpose description language that will be described later. Depending on the richness of the formal model of the artifact being designed, the system may suggest new alternatives, and also check the consistency of decisions made by the designer. Theoretically, fully automated systems could be constructed, but this is not the approach taken in this paper.

Consider the following motivating example shown in Fig.1. This example shows three design options defined by different designers to model the "*Genre*" information item in an UML class diagram modeling a CD catalogue. In Fig.1-a, the designer decided to model "*Genre*" as an *attribute* with multiplicity one or more. In Fig.1-b, the designer decided to model "Genre" as a *class* that has an association with a *CD* class, and in Fig.1-c another designer decided to model a "*Category*" information item instead of "*Genre*" to represent the same kind of information. This designer decided to model *Category* as a *class* with a self-relation of type aggregation to represent the subcategory concept.

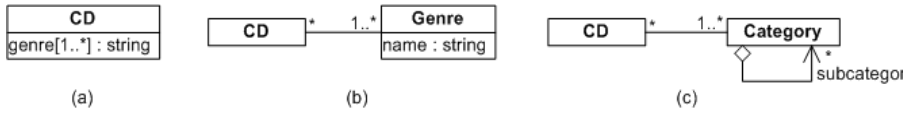

Fig. 1. Design options to model Genre item information

Since these artifacts are described in the same formal model (UML class diagrams), and refer to the same domain (CD catalogues), a fourth designer could retrieve the DR representations of these artifacts (for instance, in a distributed environment), and integrate both rationales to design a new class diagram for this domain, reusing existing (partial) solutions.

In this particular case, the fourth designer could consider the "*Genre*" and "*Category*" information items to be really the same, and thus integrate the DR representations for each. For instance, he may consider modeling "*Category*" as an aggregation, but also taking the idea of allowing multiplicity one or greater, taken from the other modeling alternative. Thus, this designer could incorporate into her design the reasoning (arguments for each alternative considered) used by the other designers, and add her own reasons as well, finally making her own decisions generating a new DR. From this point of view, both software maintenance and evolution can be considered as simply a continuation of a previous design process, captured in a given DR. This DR enables a type of reuse at the highest abstraction level, where rationales are re-employed in designing a new artifact.

In this paper, we next present a DR representation vocabulary for software designs, using the Kuaba[1] ontology. Next, we address the issue of how the formal semantics of the artifacts being described can be integrated with the design process. Next, we present the operations needed to support their reuse when designing new artifacts. We conclude by pointing out further work.

2 Kuaba: the Design Rationale Ontology

As previously mentioned, it is desirable to represent DR in a formally precise and computable way. Ontologies are good candidates for representing DR in a formally precise and computable way, since, they are knowledge representations, where a set of objects and their relationships are described through a defined vocabulary.

The Kuaba ontology describes a set of elements that express the DR domain. Our objective in proposing this ontology is to provide a vocabulary for DR described in an ontology definition language, such as F-Logic [4], that allows attributing semantics to recorded DR content, and to define a set of rules and computable operations to support the use of DR in designing new software artifacts.

The vocabulary described by Kuaba extends the argumentation structure of the Issue Based Information System (IBIS), composed by *issues*, *positions* and *arguments*. The extension enriches this argumentation structure by explicating the representation of the decisions made during design and their justifications, and integrating this argumentation structure with descriptions of the produced artifact, and with information about the design history (when decisions were made, who made them, what design method was used, etc). Fig. 2 shows the elements of the vocabulary defined by the Kuaba ontology, using the UML notation only to help visualization; some relations and constraints were hidden to simplify the presentation.

[1] "Kuaba" means "knowledge" in Tupy-guarany, the language of one of the native peoples in Brazil.

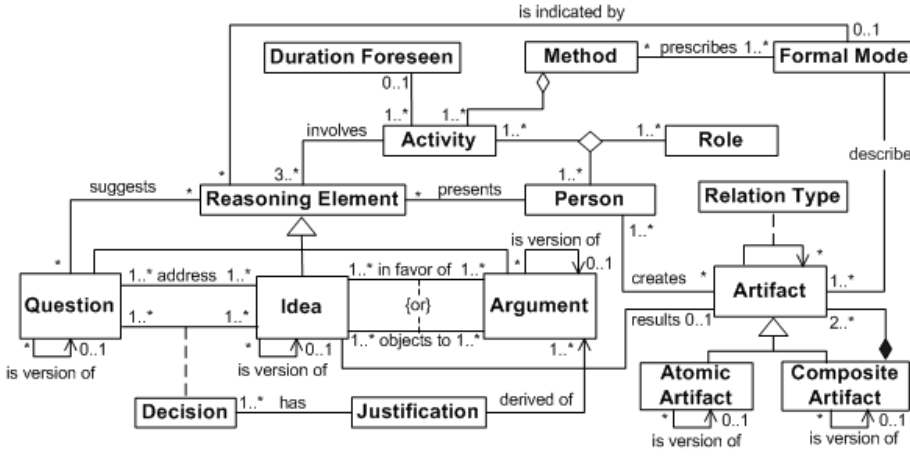

Fig. 2. The elements of the Kuaba Ontology Vocabulary

Briefly described, the Kuaba vocabulary represents the people involved in a design activity, their respective roles and the reasoning elements used for organizing and recording their solution ideas about the artifact that is being constructed. These reasoning elements represent the design problems (questions) that the designer should deal with, the possible solution ideas for these problems and the presented arguments, described according to formal model prescribed by the design method used. People involved in the artifact design make decisions about the acceptance or rejection of the solution ideas presented. Each decision must have a justification that explains the "*why*" it was made. Justification is always derived from one or more arguments. The ideas accepted during the design process originate artifacts that can be either atomic artifacts or composite artifacts. All reasoning elements and artifacts have a "*is-version-of*" relation, representing the fact that any one of them may be based on an existing element, that may be either part of a previous version of this same artifact, and therefore the design is actually evolving it, or part of a different design that is being reused in a new context.

2.1 Representing Design Rationale

Normally, the first activity done by the designer in designing a software artifact is the choice of design method or process that will be used to achieve the design. When the designer chooses a design method, he indirectly determines the formal model(s) that will be used to describe the artifact. The existence of a formal model for the artifact determines, to a great extent, the questions and ideas that the designer can propose, since they are pre-defined by this model. For example, in the motivating example in Fig.1 if the designers have chosen the Unified Process [5], the DR will be expressed in terms of reasoning elements defined by the UML formal model for class diagrams.

Fig.3 shows a graphical representation we have created to help visualizing instances of the Kuaba vocabulary, showing the portion of the design regarding the alternatives to model "*Genre*" in the motivating example.

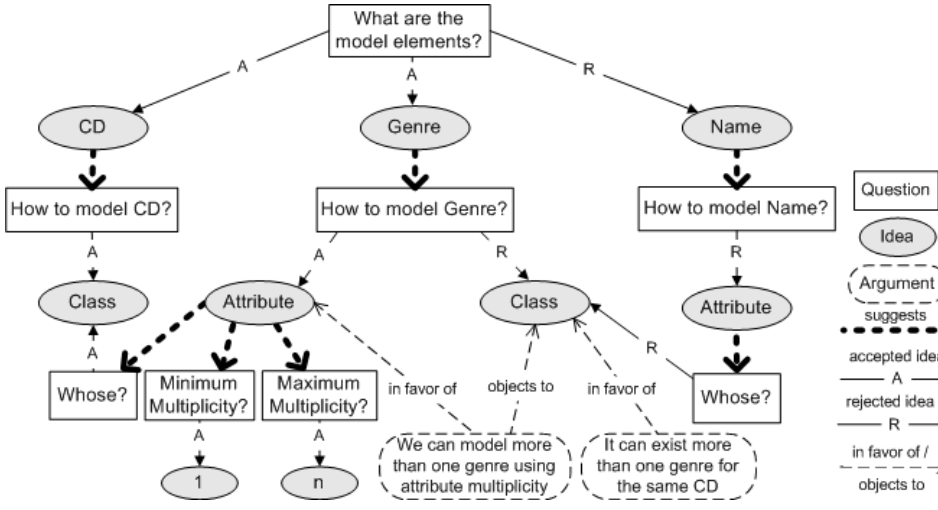

Fig. 3. A DR representation regarding "Genre" in the motivating example (Fig. 1)

Following the UML formal model, the initial question (represented as rectangles), "*What are the model elements?*", represents the first problem to be solved in designing a class diagram - the identification of its constituting elements. This question is addressed by the ideas "*CD*", "*Genre*" and "*Name*" (represented as ellipses), whose values are determined by the designer's knowledge of the domain, or were extracted from the DR of a previous phase, requirements elicitation, which is not addressed in this paper.

After establishing these first alternative ideas for the CD Catalogue model elements, the designer must decide how each one of them will be modeled using the UML, to make up the final class diagram. This next step is represented in by the "suggests" relation, which determines questions entailed by ideas – "*How to model CD?*", "*How to model Genre?*" and "*How to model Name?*". The possible ideas that address these questions are determined by the formal model for UML class diagrams – elements can essentially be a class, an attribute, or an association. Accordingly, the "*Class*" and "*Attribute*" ideas linked to the "*How to model Genre?*" node are established as an instantiation of the UML formal model. Since the "*Attribute*" idea, in turn, must be associated with a "*Class*" according to the UML model, the question "*Whose?*" is suggested, which in turn will be addressed by the idea corresponding to the class with that attribute it is. The "*Argument*" element instances represent the experiences and the knowledge that the designers are employing in the artifact design.

In addition, we also show the final decision made, indicating each alternative answer to each question with an "A" (for accepted) or "R" (for rejected) label. Thus, the example represents the fact that the designer decided to accept the "*Attribute*" alternative to the question "*How to model Genre?*", in detriment of the "*Class*" alternative. The sub-graph of the DR made up of "Question" and "Ideas" is actually an AND/OR graph [6] that can be seen as a goal decomposition of the root node, which is always a "Question". This allows us to define rules that can suggest decisions about the acceptance or not of the proposed solution ideas. For example, the

software could suggest rejecting certain ideas based in the following rule: if an idea associated with a question of type "OR" is accepted by the designer, then all other ideas associated to this question will be rejected.

3 Using the Representations of Design Rationale

The use of the DR representations requires different kinds of operations on the recorded content. The explicit and semantic representation of DR in a language formally defined and specifically designed for the description of ontologies, allows these operations to be computable by engines to support the design of new artifacts. The operations over DR representations can be grouped into *queries*, operations for manipulating an existing DR representation, and operations for integrating of two or more DR representations (Kuaba ontology instances).

The first group allows formulating relevant questions about the design process and about the produced artifact. The operations of the second group involve the creation and destruction of instances of the classes and properties defined in the Kuaba ontology. The operations for the integration of two or more DR representations involve matching different instances of the Kuaba ontology. The types of operations identified until now for the integration of two or more DR representations are: search, copy, union and substitution.

4 Further Work

Our current research includes: the implementation of the operations defined in this paper to validate the reuse of software artifacts through the integration of existing DR representations; and the investigation of the use of the Kuaba ontology to represent DR also in domains where there are no well defined formal models to describe artifacts. For an extended version, see http://www-di.inf.puc-rio.br/~dschwabe//papers/DesignRationale_and_Ontologies.PDF

References

1. Kunz, W., Rittel, H. W. J.: Issues as Elements of Information Systems. Institute of Urban and Regional Development Working Paper 131, University of California, Berkeley, CA (1970)
2. Lee, J., Lai, K.: What's in Design Rationale. Human-Comput. Interaction, No. 6 (3-4) (1991) 251-280
3. OMG: Unified Modeling Language Specification version 1.5. March (2003)
4. Kifer, M., Lausen, G.: F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance and Scheme. ACM SIGMOD May (1989) 134-146
5. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Reading, MA.: Addison-Wesley (1999) 463p
6. Nilsson, N.: Principles of Artificial Intelligence. Morgan Kaufman Publishers (1986) 476p