Design Rationale Representation in Requirements Engineering using the KAOS meta-model

Ernani Gaspar Santos[1], Adriana Pereira de Medeiros[2],

[1] Universidade Candido Mendes, Av. Anita Peçanha, 100, Pq. São Caetano 28030-335, Campos dos Goytacazes, RJ, Brasil
egsantos@acm.org

[2] Depto de Ciência e Tecnologia, Universidade Federal Fluminense (UFF), Rua Recife, 28890-000, Rio das Ostras, RJ, Brasil
adrianamedeiros@puro.uff.br

Abstract. Requirements specifications made in a poor or incorrect manner have been recognized as a source of problems in software development. Recording design rationale in this activity may contribute to a better reasoning about requirements and how to model them, since the software engineers need to carefully evaluate the justifications for their decisions. This work investigates the design rationale representation for requirements models using the Kuaba approach and the KAOS meta-model. It shows that representing design rationale taking advantage of the design meta-models' semantics can favor improvements in requirements models quality, working as design feedback. It also contributes to requirement changes management by giving semantics to requirements tracing and supporting impact analysis.

Keywords: Design Rationale; Requirements Engineering; Meta-model.

1 Introduction

Many Design Rationale (DR) approaches have been used in Software Engineering since the late '80s [1], such as DRL (Decision Representation Language) [2] and RATSpeak [3]. Regardless of the approach, DR is the description of the reasoning employed to determine the design of an artifact. It typically includes explanations of the alternatives for solving design problems, the reasons behind the decisions made regarding the alternative that best solves them and which options were rejected. Although there are many different DR approaches, capturing and representing it remains a challenge in Software Engineering, especially in Requirements Engineering (RE) due to the volatility and high level of abstraction of domain concepts, usually obtained from statements in natural language.

During the RE activity the domain provides to software engineers those concepts related to the real world upon which the software will operate, and its environment, bringing value to their users. These concepts represent abstractions that must be expressed in a way that preserves its semantics. The meta-models, from which the models are obtained, usually meet this requirement whenever they are rich enough to represent these abstractions, maintaining meaning and creating a bridge to implementation artifacts. In RE the meta-model provides the semantics for the representation of abstractions related to the stakeholders needs, whether functional or non functional, leading to the Requirements Model. Requirements specifications performed in a poor or incorrect manner have been recognized as a source of problems in software development, making the RE critical in software process, since all development is based on knowledge gained in this activity [4]. Recording DR in this activity may contribute to a better reasoning about requirements and how to model them, since the design alternatives and the decisions are carefully evaluated.

Usually, the semantics provided by the meta-model used to describe the artifacts is not exploited in the DR approaches reported in the literature, such as IBIS - Issue Based Information System [5] and DRL. The content of the DR produced using these approaches are mostly informal and incomplete. This prevents the computational processing of such knowledge and its use to support the design of new artifacts. In these approaches the knowledge applied to the design is not represented in a standardized way, since they generally do not incorporate the design meta-model's semantics. The recorded DR also lacks common properties, since a common taxonomy is not used in its description. This fact precludes its use in a comparative way, since the DR contents, manually expressed using text and concepts of different taxonomies, chosen by the designers, are not formally equivalents. In this case, one can only use the recorded DR for each model, separately.

Kuaba [6] is an argumentation-based approach that incorporates the semantics provided by the design meta-models in the DR representation. The use of this semantics formalizes and enriches the DR contents, allowing their computational processing to support the design of new artifacts. The Kuaba approach has been used so far to represent DR in conceptual modeling (analysis), using the UML meta-model [7], and navigational modeling for Web applications (design) using the meta-model of the OOHDM - Object Oriented Hypermedia Design Method [8]. Requirements are notoriously volatile, the product of a negotiation process inherently difficult, usually because they represent different interests declared ambiguously. Therefore, the DR representation during the requirements modeling can be a more difficult task than in those (conceptual and navigation modeling), where the level of abstraction and susceptibility to change is usually lower. This work investigates the usage of the KAOS meta-model [9] in the Kuaba approach to represent the knowledge used by software engineers to model requirements, adding thus its semantics to the recorded DR. It also shows that this approach can favors improvements in requirement models quality, since careful argumentation must be done for each possible solution for the artifact design, working as a design feedback. In addition, it discusses how the approach can contribute to requirement changes management by giving semantics to requirements tracing and supporting impact analysis.

The Kuaba approach and the KAOS meta-model are presented and discussed in Section 2. Section 3 discusses the investigation framework and planning, showing how the requirements design tests were conducted, also presenting and discussing the DR representation examples. Conclusions, contributions and future works are presented in Section 4.

2 Kuaba Approach and KAOS Meta-Model

Kuaba is an argumentation-based approach for representing DR in model-based designs. Its main purpose is to allow DR computational processing to support design reuse, particularly in software designs. Model-based design is a category of design problems that can be viewed as a process of instantiating a meta-model. This meta-model represents the semantics used to describe the produced artifacts. Kuaba differs from other DR approaches, such as IBIS and DRL, because it incorporates the semantics provided by the design meta-models to the DR representation, generated from its ontology instantiation. Fig. 1 illustrates part of the Kuaba ontology vocabulary. The diagram is presented as a UML class model to aid visualization.

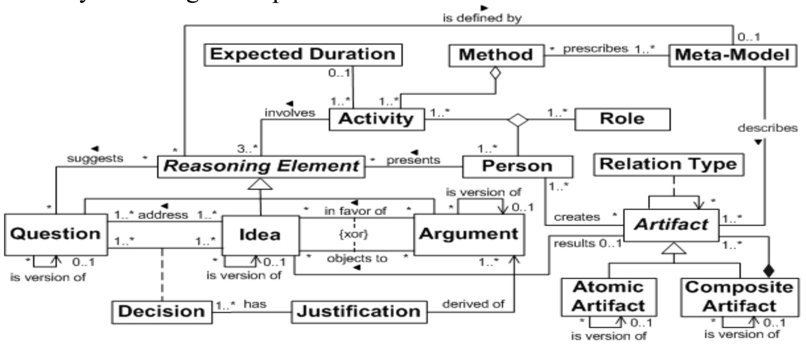

Fig. 1. Kuaba ontology vocabulary.

An artifact design involves a series of reasoning elements that include questions related to the design problem, the solution ideas addressing these questions and the arguments for or against the ideas presented. The “is version of” relationship defined for the elements “Question”, “Idea” and “Argument” allows representing that a reasoning element is obtained from the DR of another design. This instance may be of an earlier version of the design, which is being used to improve the artifact design, or from a different design that is being reused in a new situation. The acceptance or rejection of an idea as a solution to a design question is registered by the “Decision” element. A decision must have a justification for the acceptance or rejection of an idea proposed and is always derived from one or more arguments. The Kuaba ontology vocabulary also includes elements to represent information about the artifacts produced from the accepted ideas, the design method applied and the meta-model prescribed by it to describe these artifacts. The current version of ontology and its instances are represented in OWL - Web Ontology Language [10].

The usage of the design meta-model’s semantics proposed by the Kuaba approach allows representing DR in a standardized way and consistent with the chosen design method. The questions and design ideas that form the DR structure are defined using the concepts established by the meta-model prescribed by the design method. So, the DR of artifacts produced in different projects by different software engineers, but using the same meta-model, has a common structure of questions and ideas. It also allows performing inferences and computable operations to support the use of recorded knowledge in the design of new artifacts. An example of these operations is the DR integration, in which DR related to different artifacts designed with the same meta-model and representing the same domain can be automatically combined to produce new artifacts [6]. This integration enables the design reuse in a higher level of abstraction, since the designer can initiate a new artifact design from the resultant DR, accessing a larger set of solution alternatives and taking advantage of the knowledge and experience of other designers, recorded as arguments and justifications. In addition, the meta-model's semantics usage as part of the DR approach enables the semi-automatic capture and representation of this knowledge in design tools, since many questions and design ideas that form the DR structure can be obtained from the design meta-model used by the tool. When all DR structure (questions, ideas and arguments) is represented manually, as occurs in the majority of DR tools such as Compendium[11] and SEURAT [3], each designer can record the questions and the solution alternatives using their own terms, not considering the taxonomy defined by the meta-model or its restrictions. This makes the computational processing of DR difficult, allowing only queries to the recorded knowledge. Furthermore, it entails a considerable time and cost increase in a software project, making the designers desist from using these tools and, consequently, recording DR. A software design tool, called KSE - Kuaba Software Engineering, is being developed as an extension of the ARGOUML to support the semi-automatic capture and representation of DR using the Kuaba approach. In its current version, KSE automatically generates part of the DR structure (questions and design ideas) during the design of models represented by UML class diagrams.

The history of meta-modeling in RE shows a semantics enrichment process in meta-concepts, since RML - Requirements Modeling Language [12] to GRL - Goal Representation Language[1] [13]. The goal-oriented meta-models present semantics that can represent abstractions of the RE complex concepts. The KAOS meta-model was chosen for the modeling tests in this research because it has been used in several projects in Europe and is stable enough to support DR recording in the RE. This meta-model includes four models: Goal Model, Object Model, Responsibility Model and Operation Model. Fig. 2 illustrates the complete meta-model in its own graphical representation. The models are related through associations between concepts of different sub-models. For example, an agent may be responsible for "Requirement" instances, and "Entity" instances are inputs for or outputs from operations.

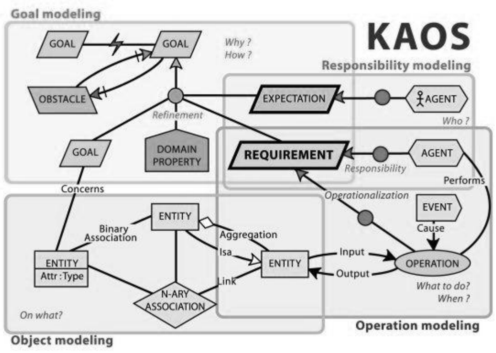

Fig. 2. The four models of KAOS meta-model [14]

[1] http://www.itu.int/rec/T-REC-Z.151/en

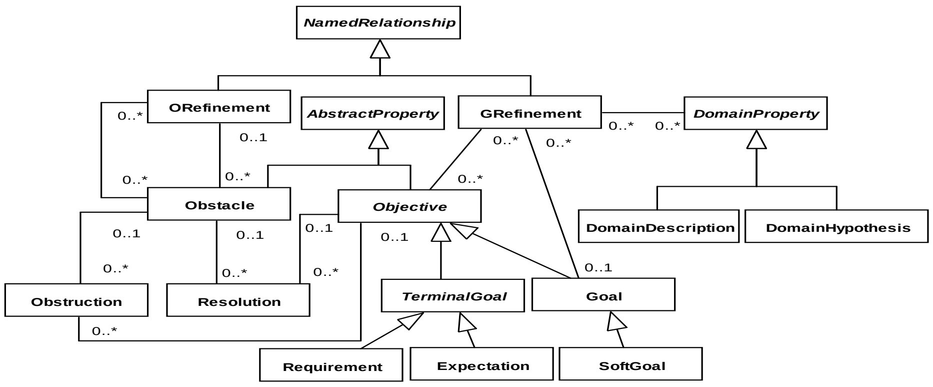

Fig. 3. KAOS meta-model concepts for the Goal Model [14].

During the Goal Model design, goals are refined according to a defined discovery strategy, from the root goal unto indivisible concepts. Refinements can be either "AND" or "OR". "AND" refinement types are satisfied when all goals refining some goal are satisfied. Otherwise, in "OR" type refinements, satisfying just one refining goal is enough. The Object Model is a conceptual model that is very similar to an UML class model for domain classes. The Operations Model describes the behavior of agents in order to realize their responsibilities in satisfying requirements or expectations while performing operations. In other words, requirements are satisfied through "operationalization". The sub-models answer questions about the system (How, Why, Who, What to do, When and Upon what) [14]. The models that arise as answers to these questions show the rationale of one possible design instance of the problem in hand. DR, otherwise, shows the reasoning about all the possible design alternatives envisioned by requirements engineers during such a task, where each of them is analyzed thoroughly as a possible design solution.

The partial KAOS meta-model is illustrated in Fig. 3, showing the concepts of the Goal Model using UML notation. Ideas from Kuaba ontology vocabulary are instantiated using the meta-model's concrete classes. The concepts describing relationships and attributes within the meta-model are usually instantiated as questions. For instance, the concepts Goal, GRefinement (goal refinement), Obstacle, Softgoal, Expectation and Requirement are used to define ideas, and Obstruction and Resolution, along with refinements relationships, are used to define questions in the DR representation (Fig. 4). In this way the semantics of the meta-model, or the navigational possibilities in that semantic network, is incorporated to represent reasoning elements in the DR for Goal Models. These elements can be automatically generated from the meta-model in a design tool.

3 DR in Requirement Engineering using Kuaba and KAOS

In software development, the definition of the design problem must be obtained from a set of disconnected knowledge and intentions. Software engineers, together with other stakeholders, have the mission to collect this information and define the problem to be solved, as well as formulate its solution. The DR representation during the requirements modeling creates the possibility of using this knowledge in RE tasks.

The examples studied in this research, although not exhaustive, are intended to simulate real design tasks undertaken by software engineers in RE. They are designed to explain how Kuaba ontology instances (DR) are created incorporating the KAOS meta-model's semantics for representing DR. The framework of the investigation was based on two major constructs: (a) simulate a real requirements modeling task of a software engineer and (b) design a set of modeling tests that could exploit the DR representation for requirement models using the most important KAOS concepts in all of its sub-models. This framework allows to verify how the captured DR can support the RE activity as well. Although the framework covers all the research work, this section discusses mostly goal models due to their relevance within goal-oriented paradigm, and also because of size constraints of this paper.

The design test fixtures were divided in three categories. The first one uses the usual approach of representing DR during the modeling task. In this approach the best solution for the artifact is chosen and its respective DR is recorded. The process of argumentation to justify this choice leads the software engineer to evaluate other solution possibilities, considering the design options available in the meta-model. Therefore, DR brings a careful and thoughtful approach to modeling. The second category is about first modeling and, after finishing the modeling task, recording DR. The software engineer works on the DR representation as if documenting the reasoning used in the choices for the artifacts produced. In this case, part of the knowledge invested in the modeling can be lost, since the software engineers may not remember all their reasoning. The third category is about don´t model at all, or modeling using a different approach, just representing DR directly from stakeholders reported needs. This situation presupposes a deep understanding of the design meta-model used, since it goes directly to instantiate Kuaba ontology using the meta-model semantics.

In order to simulate real modeling sessions, a list of statements about a library system domain was chosen [15]. For the purpose of this paper, only the item "***The Library lends books and magazines, which must have been catalogued, for registered users***", which is the main process that will be supported by the system, is used. Fig. 4 graphically illustrates the DR recorded during the design of a partial goal model for this library domain.

The DR representation is always initiated by the question (represented as a rectangle on top of Fig. 4) which asks about which domain concepts are known. The domain ideas addressing this question are shown as ellipses just below it. These domain ideas suggest questions of how to model those concepts. The design ideas considered as possible solutions to model them are derived from the KAOS meta-model's semantics (Fig. 3). According to the Kuaba approach, all design options available in the meta-model can be considered to address this question in the DR representation. However, for a better visualization, just some of these ideas are illustrated in Fig. 4. For instance, to address the question "How to model this concept?" suggested by the domain idea "Loans of books and magazines are made" are illustrated only the ideas "Goal" and "Goal Refinement".

The meta-model defines that a goal refinement is also a goal. Navigating the meta-model illustrated in Fig. 3, it can be observed that the refinement association has two endings. They are expressed as the questions in the DR representation: “Which is the refinement?” and “Refined from which?”. In this case the idea “Loans of books and magazines are made” is accepted as a goal and as a refinement of “The library lends books and [...]”, as other ideas, if shown in this illustration, would be rejected. Justifications (not illustrated in Fig. 4) for these decisions are also recorded and must take arguments into account. There are still questions regarding the type of refinement (AND/OR) in order to define the rule of the goal fulfillment. Concerning the ideas above, in Fig. 4 the question about which is the refinement termination is addressed by that goal itself and the question about which goal it is refined from is addressed by “The library lends books and [...]”. The arguments given for the refinement type take into account that this goal, as well as others that refine this goal, must be satisfied for this one to be fulfilled. Thus, the decisions is to accept the “AND” type and reject the “OR” type.

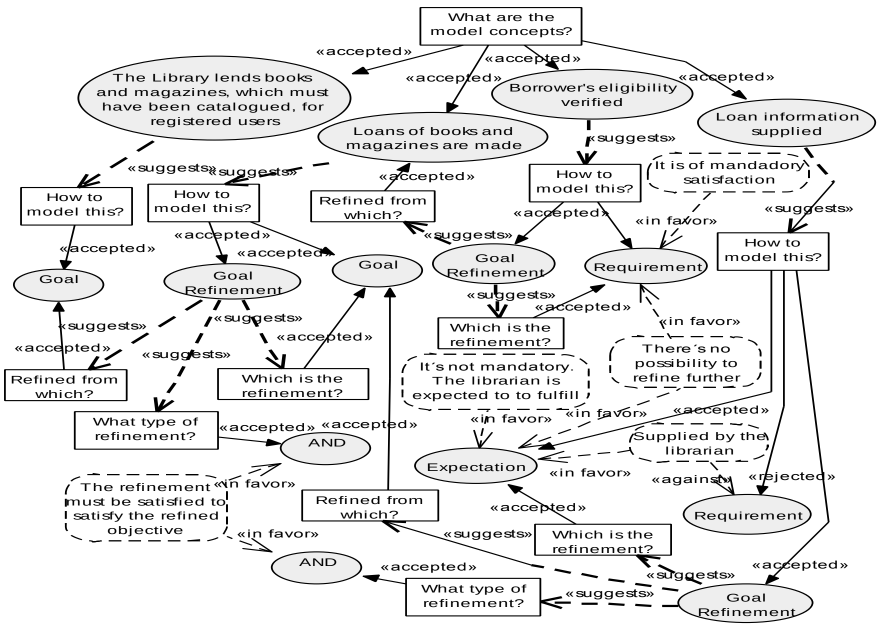
What are the model concepts?
«accepted»
The Library lends books and magazines, which must have been catalogued, for registered users
Borrower's eligibility verified
Loan information supplied
Loans of books and magazines are made
«suggests»
How to model this?
It is of mandadory satisfaction
Refined from which?
Goal
Goal Refinement
Requirement
«in favor»
Which is the refinement?
There´s no possibility to refine further
It´s not mandatory. The librarian is expected to to fulfill
What type of refinement?
AND
Expectation
Supplied by the librarian
«against»
«rejected»
The refinement must be satisfied to satisfy the refined objective

Fig. 4. Design rationale representation for refinements in a Goal Model.

Goal refinements may be terminal, indicating the end of the refining process, also according to the meta-model shown in Fig. 3. In this case the ideas “Requirement” or “Expectation” should address the question “Which is the refinement?”. The semantic difference between the concepts of requirements and expectations is significant. Requirements are the responsibility of software agents, having their satisfaction guaranteed. By contrast, expectations are the responsibility of environment agents, as humans are classified in the KAOS framework, meaning that agents are expected to fulfill their responsibility, although this is not guaranteed. The verification of the borrowers’ eligibility, which is shown just below the root question in Fig. 4, must be guaranteed. This implicates that the idea of a requirement to model this concept is accepted, in contrast with "Loan information supplied” that is modeled as expectation, since it can be registered by the user or not. This is also explained by the arguments shown in this example. It can be noticed that these concepts, requirements and expectations, are not refined further. These concepts are represented in Fig. 7 as parallelograms with thicker contour lines. Requirements are shown in light grey and expectations in a darker tone.

The obstacle analysis is an important task during the goal modeling with KAOS, in which one is able to foresee situations that obstruct or impede goal satisfaction. These situations might be related to architectural, implementation or even functional design. The obstacle analysis reveals technical risks for the project, which may be solved or worked around still in the RE activity, bringing benefit to the whole development. The strategy whenever modeling with KAOS is to test every goal or requirement to look for obstructions. With the aid of DR, the arguments may help to identify these obstructions easier, since the arguments mean the reasoning about possible solutions and therefore may entail discussions about obstructions to that proposed solution.

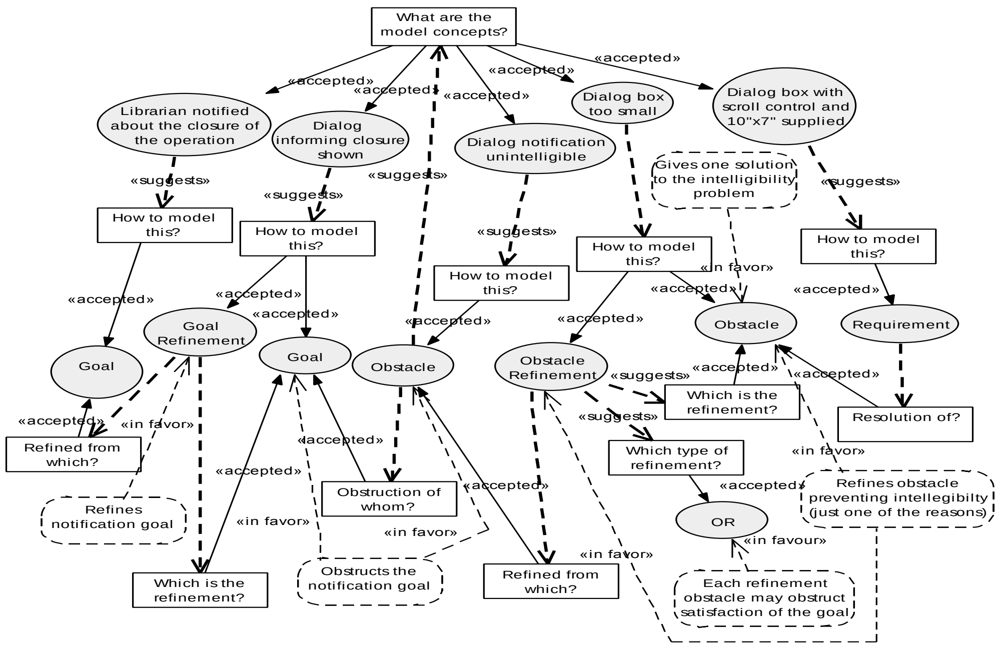
What are the model concepts?
«accepted»
Librarian notified about the closure of the operation
Dialog informing closure shown
Dialog notification unintelligible
Dialog box too small
Dialog box with scroll control and 10"x7" supplied
«suggests»
How to model this?
Gives one solution to the intelligibility problem
«in favor»
Goal
Goal Refinement
Obstacle
Obstacle Refinement
Requirement
Resolution of?
Refined from which?
Which is the refinement?
Which type of refinement?
Obstruction of whom?
OR
«in favour»
Refines notification goal
Obstructs the notification goal
Refines obstacle preventing intellegibilty (just one of the reasons)
Each refinement obstacle may obstruct satisfaction of the goal

Fig. 5. Design Rationale Representation for the Obstacle Analysis.

Fig. 5 shows the reasoning about the situation where the satisfaction of the transaction closure notification is obstructed. It is caused by the unintelligibility of the presented closure message. The problem may relate to many situations like the contrast of text and canvas colors, or the box may be too small. It is important to notice that the obstacle refinements are of type “OR”, which means that if any of them holds, the goal satisfaction condition would be compromised. The DR of Fig. 5 does not show all these obstacle ideas in order to prevent cluttering the illustration. The relationship “suggests” from the obstacle idea to the root question means that the designer at the time of this obstruction discovery needed to add new domain ideas to cover for the situation. These ideas prove to be proposed solutions for refinement of the obstacle itself and its resolution, shown from the top center to the right of Fig. 5. This example also poses a new experience about the DR representation and KAOS meta-model semantics, showing that obstacles may also be refined, as goals are, and their relationships with goals and requirements represented as question: “Obstruction of whom?” and “Resolution of?”. The requirement presented that addresses the size and features of the dialog box is the resolution for the obstruction.

The evaluation of the tests showed that the DR representation and the requirements model itself are more complete when working in parallel. This happens because no argument is overlooked, even those apparently thought obvious or common sense are captured when DR is recorded during the modeling. Each new element added to the model leads to reasoning about the arguments and the decisions, which in turn can result in a revision in the model itself, thus creating cycles of improvement during modeling and DR representation process. During this first situation, which is capturing the DR during the modeling task, it was noticed that the requirement “Borrower’s eligibility is verified” is neither complete nor correct. The DR of the original partial goal model of Fig. 4 shows that the “requirement” idea associated to the question about “How to model this concept” is an accepted idea due to the argumentation in favor of this solution. However, a more thorough analysis of the domain idea and its arguments presented that this solution is more generic than needed, since there is a declared need stating that “The borrower must be registered to make loans”, which is not taken into account. This situation implies changes in DR, given that “Borrower´s eligibility verified” actually may be refined further, thus being modeled as a goal. Fig. 6 shows these changes of goals and requirements although it illustrates an Operations Model. The verification and validation using DR considers the recorded arguments and the all proposed solutions to aid the decisions about completeness and correctness issues. About the DR recording done after the model has been finished, it was observed that the representations are poorer and the models may have errors, because there is no way to remember every argument and solution alternatives that might have been envisioned at the time of modeling, even though the recording is held almost immediately after completion of the model. Another noticeable aspect is that when the software engineer has sufficient knowledge of the design meta-model’s semantics, the DR can be produced without building the model in advance. This practice led to more complete representations, i.e., with careful argumentation, better choices and greater detail. In this case, the model itself could be computationally obtained from the DR, since it uses the design meta-model’s semantics, according to the Kuaba approach.

3.1 Design Rationale in Requirements Evolution

It is accepted that requirements are volatile and the impact of changes can be very large depending on the project stage. Therefore, supporting the management of the requirements evolution, allowing the tracking and evaluation of changes, and measuring the impact they may have on the project, is an important contribution to RE. Tracing artifacts usually takes into account a dependency between artifacts with limited semantics, which may limit the effectiveness of impact analysis. Once identified the source of change, namely, that certain requirement will be changed, the trace just indicates the amount of artifacts that may be changed in each development activity. In mechanisms for versioning control, the log recorded is a textual description of the changes in natural language. The DR representation with Kuaba takes into account the decisions made during the design of artifacts, based on arguments and justifications that reflect the reasoning of software engineers about the problem domain and the use of the meta-model concepts to express the abstractions of this domain. These arguments may support the impact analysis through the possibility of using semantic elements in the analysis. Moreover, as DR representations have common structure and semantics provided by the ontology and the meta-model, they can be compared and measured, as well as the impact of requirements changes through tracing mechanisms of their own semantic network (DR representation).

Fig. 6 shows changes in the DR for the operations models provoked by a new need posed by stakeholders during a business process revision. This change adds a new rule to borrower verification, stating that borrowers must not have overdue loans to be able to make others. The accepted requirement solution for the idea "Borrower's eligibility is verified" was changed to goal due the argument about the borrower's existence in the database. This reasoning introduces the accepted solution that there is a new requirement refining the changed goal. This illustration also shows that a new need is brought to attention. As the refining structure of the former need is already in place, the new idea is introduced and accepted as a requirement refining "Borrower's eligibility is verified". This is illustrated in Fig. 7 that shows the partial KAOS goal model with the three concepts changes (center to bottom). Fig. 6 shows the impact in the DR representation of the operations model and the evolution when the operation "Verify borrower's eligibility" is deleted, because the KAOS's semantics does not allow "operationalization" of a goal concept. "Verify borrower existence in database" and "Verify overdue loans […]" operations are added "operationalizing" the also new requirements added in the goal model. This means that these operations make possible the satisfaction of those requirements through the performance of a software agent "Loan Transaction Component", not show in the illustration.

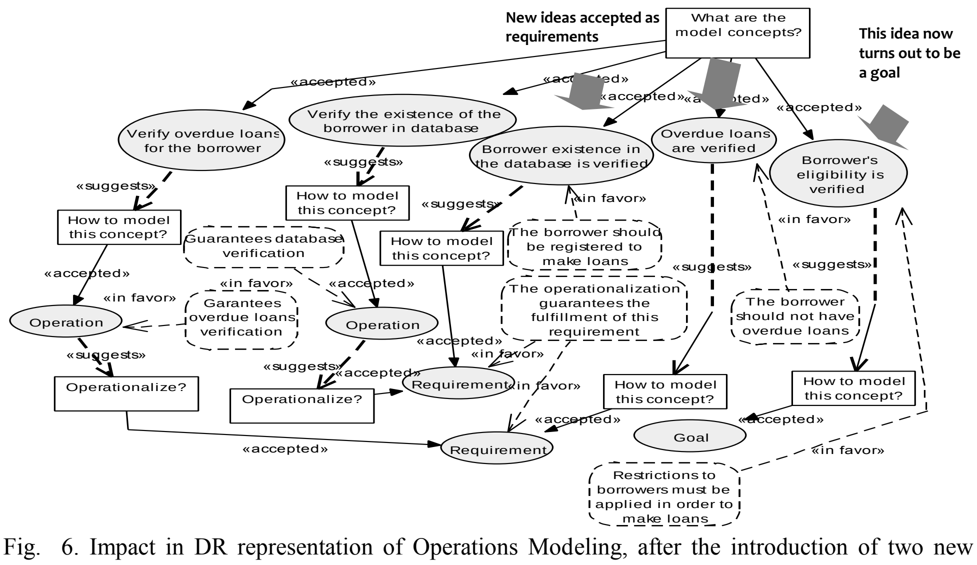
New ideas accepted as requirements
What are the model concepts?
This idea now turns out to be a goal
«accepted»
«accepted»
«accepted»
«accepted»
«accepted»
Verify overdue loans for the borrower
Verify the existence of the borrower in database
Borrower existence in the database is verified
Overdue loans are verified
Borrower's eligibility is verified
«suggests»
«suggests»
«suggests»
«suggests»
«suggests»
How to model this concept?
How to model this concept?
How to model this concept?
How to model this concept?
How to model this concept?
«in favor»
«in favor»
The borrower should be registered to make loans
Guarantees database verification
«accepted»
«accepted»
«in favor»
«in favor»
Garantees overdue loans verification
Operation
Operation
The operationalization guarantees the fulfillment of this requirement
The borrower should not have overdue loans
«suggests»
«suggests»
«accepted»
«in favor»
Operationalize?
Operationalize?
«accepted»
Requirement
«in favor»
«accepted»
«accepted»
Goal
«accepted»
Requirement
«in favor»
Restrictions to borrowers must be applied in order to make loans

Fig. 6. Impact in DR representation of Operations Modeling, after the introduction of two new domain ideas, related to quality and new stakeholders' needs, were introduced in the Goal Model.

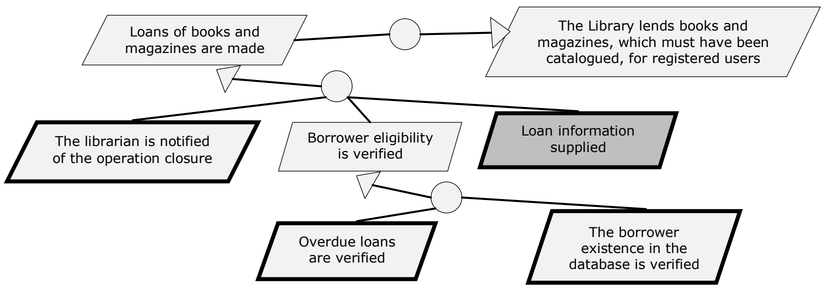
Loans of books and magazines are made
The Library lends books and magazines, which must have been catalogued, for registered users
The librarian is notified of the operation closure
Borrower eligibility is verified
Loan information supplied
Overdue loans are verified
The borrower existence in the database is verified

Fig. 7. Design revision caused by quality improvements and new stakeholders' necessities.

DR offers another way to perform the analysis of changes, since the analysis is made over recorded knowledge in a formal language (OWL) incorporating the semantics of the meta-model used to describe the artifact. Moreover, the visual analysis of recorded DR also offers a way to devise the mechanism for impact analysis, as can be observed in the presented examples.

4 Conclusions

The DR application in Software Engineering and RE has been an active area of research for the past two decades [1]. However, the complexity and the vastness of the subject, ranging from RE to cognitive science, limited its evolution in this period. It reflects the difficulties and the size of research space. Specifically in RE, most of the research work is more focused on negotiation processes and initial activities of software development, than in the techniques used in requirements modeling and specification. The research presented in this paper focuses on the DR representation for the requirements modeling incorporating the design meta-model's semantics, according to the Kuaba approach. The goal is investigating how the explanations about design decisions made in requirements modeling (DR), can contribute to the practice of RE. It contributes to the research field, since that the usage of the design meta-models' semantics in the DR representation in RE has not been addressed previously.

The tests performed in this work simulate software engineers' day-to-day modeling tasks and show how DR can be represented during the requirements modeling incorporating the semantics of the KAOS meta-model. This means that the knowledge and the experience of the software engineers employed in the requirements modeling can be represented in the DR using the Kuaba approach, even though the DR representation in this activity seems to be a more difficult task than in those (conceptual and navigation modeling) where the approach already was tested.

A general aspect observed during the tests is that the recording of DR for software requirements design favors the quality improvement of the models created, since the meta-model's semantics is taken into account when Kuaba approach is used. This approach avoids syntax mistakes and prevents semantics inaccuracies in the requirements specification. In addition, it leads to a careful analysis of all modeled concepts as they are scrutinized by argumentation for or against until a decision is made. Moreover, as DR representations have a common structure and semantics provided by the ontology and the meta-model, they can be computationally compared and measured. Also, they can be used to support the impact analysis of requirements changes through tracing mechanisms or their own semantic network (DR), as they preserve not only the decisions and justifications related to the chosen design solution, but their design history.

A relevant future work is to carry out a case study involving a group of software engineers using measuring criteria for quality and productivity in order to evaluate their work involving the DR representation with Kuaba. These results can contribute to a better evaluation of the analytical observations obtained in the present work about improvements in the models due to the usage of the Kuaba approach to represent DR. Adapting the design tool that is being developed to support the semi-automatic capture and representation of DR from the KAOS meta-model is another important future work. It can make the achievement of the case study viable. The DR representations produced by this design tool are generated in the OWL formal language. So, the application of the operations implemented in [6] using these DR representations to assess the possibility of reusing requirements models from the recorded DR is also an interesting work. It could ratify the observations made in this research on the use of DR to support the requirements evolution.

References

1. Dutoit, A.H., McCall, R., Mistrik I., Paech, B.: Rationale Management in Software Engineering: Concepts and Techniques. In: Rationale Management in Software Engineering. Springer-Verlag, pp. 1--48 (2006)
2. Lee, J., Lai, K.: What's in DR? In: Hum-Comp Interaction,vol. 6,no. 3, pp. 251--280 (1991)
3. Burge, J., & Brown, D. C. "An Integrated Approach for Software Design Checking Using Rationale", Design Computing and Cognition, Kluwer Acad. Publishers, 557- 76 (2004)
4. Lamsweerde, A.: Requirements Engineering. John Wiley & Sons Inc. (2009)
5. Kunz W., Rittel H.W.J.: Issues as Elements of Information Systems. Inst. of Urban and Regional Dev. Working Paper 131, Uni. of California, Berkeley, CA, USA, (1970)
6. Medeiros, A., Schwabe, D.: Kuaba Approach: Integrating Formal Semantics and Design Rationale Representation to Support Reuse. Artificial Intelligence for Engineering Design, Analysis and Manufacturing. 2008-22, pp.399--419. Cambridge University Press (2008)
7. OMG: Unified Modeling Language 2.0: Infrastructure and Superstructure. (2006)
8. Scwabe, D. and Rossi, G.: An object-oriented approach to Web-based application design. In: Theory and Practice of Object Systems (TAPOS), pp. 207--225 (1998)
9. Dubisy, F., Lamsweerde A., Dardenne, A.: The KAOS Project: Knowledge Acquisition in Automated Specification of Software. Proc. of the AAAI. Spring Symp. Series USA (1991)
10. W3C: Web Ontology Language Reference. February. (2004)
11. Conklin, J., Selvin, A., Buckingham, S. S., Sierhuis, M. "Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS", In: LAP'03: 8th International Working Conference on the Language - Action Perspective on Comm. Modeling, Tilburg (2003)
12. Greenspan, S. J.: Requirements Modeling: A Knowledge Representation Approach to Software Requirements Definition. Ph. D. Thesis - University of Toronto, Canada. (1984)
13. IUT-T: International Union for Telecommunication. IUT-T Z.15 Recommendation. (2008)
14. Respect-IT: KAOS Tutorial. Version 1.0. (2007)
15. Erikson, H. and Penker, M.: UML Toolkit. John Wiley & Sons, Inc., 1st Edition. (1998)