AN APPROACH TO HYPERTEXT-BASED REQUIREMENTS SPECIFICATION AND ITS APPLICATION

Hermann Kaindl

Siemens AG Österreich, PSE
Geusaugasse 17, A—1030 Vienna, Austria
E-mail: kaih@siemens.co.at

KEY WORDS: Hypertext, semiformal representation, requirements engineering.

ABSTRACT: Specifying the requirements of a new system to be built is one of the most important parts of the life cycle of any project, but its support in practice is still insufficient. Since pure natural language has its disadvantages, and immediate formal representation is very difficult, a mediating representation is needed. Therefore, we used *hypertext* technology to develop a novel method for requirements specification (with tool support). This approach provides both for a mediating representation during *incremental formalization*, and for convenient interaction of the requirements engineer with a computer. We have applied our approach in real-world projects, and our experience suggests its usefulness.

1 INTRODUCTION

Specifying the requirements of a new system to be built is one of the most important parts of the life cycle of any project. In the field called *requirements engineering* many approaches have been proposed [2], but the support in practice is still insufficient.

While from a theoretical point of view it would be desirable to have *formal* representations of requirements, in practice unstructured natural language is often used *informally*. There is no doubt that formal approaches are very important. However, people do not find formality helpful except where the issues are thoroughly studied. Consequently, there is a big gap between the informality in the real world and the formality finally required in computer representations.

Our approach attempts to bridge the gap in providing *semiformal hypertext* representations. Therefore, our approach and the tool supporting it are named RETH (Requirements Engineering Through Hypertext). We do not attempt to exclude or replace formal representations, but try to provide means for gradually developing and to complement them. RETH provides both for a mediating representation during *incremental formalization* [8, 9], and for convenient interaction of the requirements engineer with a computer.

Our method and its supporting tool have been applied in several real-world projects both outside and inside of Siemens. We selected three projects from three different organizations for summarizing our experience: a project for building a generic distributed control system at CERN (Conseil Europèen pour la Recherche Nucléaire) in Geneva; a project for building a mission planning system at ESOC in Darmstadt, the German branch of the European Space Agency; and a project within Siemens in Vienna dealing with requirements from the ÖBB (the Austrian railway organization).

First, we sketch the architecture of our hypertext-based tool. We then describe RETH's support for activities during requirements specification, and summarize our experiences with RETH. Lastly, we relate RETH to existing work.

2 OVERVIEW OF OUR TOOL RETH

For a better understanding of the architecture of our tool, we sketch first the underlying hypertext level. Our hypertext approach is similar to the one described and used by Kaindl and Snaprud [7] for *knowledge acquisition* in the course of building knowledge-based (expert) systems. Analogously, we let the user define explicit *partitions* of a hypertext node, that together cover the whole node. The idea is to support the user in partitioning the textual content in a machine recognizable form, serving as an additional means of introducing more formality.

In order to support convenient navigation we use a

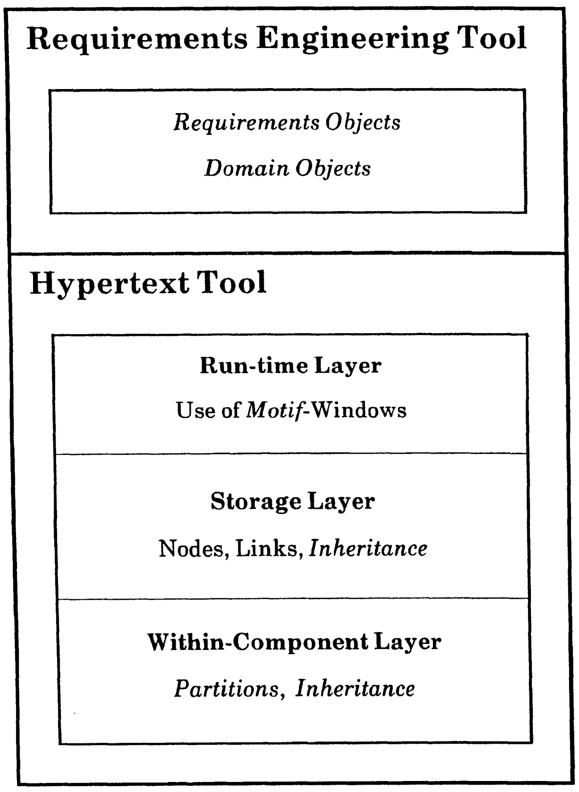

Figure 1: Architecture of RETH.

kind of bi-directional link. Moreover, we also permit links *into* nodes. Our feature of explicit partitioning of hypertext nodes supports this, and in effect these links point to partitions. In addition, taxonomic relationships can be directly represented, that support the *inheritance* mechanism.

Figure 1 illustrates the layers of our hypertext approach according to the Dexter reference model [5]. The *within-component* layer includes the partitions. The inheritance mechanism relates to both this layer and the *storage layer*. The *run-time layer* uses *Motif* for Sun Workstations, and for a more recently developed PC version it uses *Microsoft Windows*.

Our requirements engineering tool is architecturally based upon a hypertext tool we built according to this approach. While much of the interaction between users and the system is directly via the hypertext tool, the requirements engineering tool can be viewed as a level above it (see Figure 1). It specifically supports some aspects of the method as described below. Generally, every requirement object (class) and every domain object (class) is represented in one hypertext node each.

3 REQUIREMENTS SPECIFICATION USING RETH

While our approach should be generally useful, we want to specifically support requirements specification in the context of *object-oriented* approaches [3]. Therefore, we support the formation of OOA (object-oriented analysis) models. When developing OOA diagrams, we use mediating hypertext representations that include textual representations in natural language. Hypertext nodes represent (potential) domain objects and their classes semiformally.

The structure inside (*attributes*) is described in *partitions* of these nodes. Relationships/*associations* (and in particular *aggregation* relationships) are represented by use of partitions and hypertext links. The representation of taxonomic relationships between class nodes in our hypertext tool provides an inheritance mechanism. In addition, the hypertext representation allows the embedding of an interlinked *data dictionary* into the domain model.

While most OOA methods (for a comparison see [3]) focus on the creation of diagrams, textual descriptions are typically maintained separately. Our approach focuses on structuring and linking the text, but actually we combine the use of text and diagrams. This can be compared to the mixed external representations of Guindon [4].

Since we want to make full use of object-oriented principles, we also model requirements as objects in our approach. The representation of a requirement should not just be plain text, but include links to the nodes representing domain objects (as a statement *about* the domain). This leads to a tight combination of the requirements with the domain model.

While installing links is also manually possible in RETH (with an efficient user interface), there is machine support for installing links semi-automatically. Whenever the editing of a node (or partition) is finished, a parser scans the text searching for object names (using a *thesaurus*). However, such a link will be just proposed to the user, and only after she or he acknowledges the link will actually be installed.

Moreover, the representation of a requirement may contain links to other nodes representing requirements, making dependencies between these requirements explicit. In particular, relationships between *functional* and certain *quality* requirements are important since the latter can be viewed as constraints on the former.

Another very important relationship between requirements exists between functional requirements and *scenarios* (behavioral requirements). Each functional requirement can describe the purpose of one or more scenarios, and each scenario can have several purposes. Therefore, this can be in general a many-to-many association between these requirements objects.

4 EXPERIENCE WITH RETH

According to our experience in real-world projects, all the features of our method and its supporting tool were useful to some extent. In fact, some of them were worked out in detail in the course of these applications. In the following, we focus on the lessons learned in three selected projects.

A general observation is that it has been very useful to have a metamodel in mind of how to classify information. For example, when a scenario described by the potential user cannot be related to required functions, this indicates a missing function. Vice versa, when a function is requested in isolation, asking for a related scenario may reveal more information about other required functions. In this way, our method has helped to raise the right questions when something seemed missing. This improves the *completeness* of the requirements.

In all the projects, classification of the initially given requirements was a major issue. In particular, in the project at CERN the clean distinction in RETH between functional and quality requirements led to a clearer view and helped to identify some available text as specific quality requirements, and to link them to the corresponding functional requirements. More importantly, however, it turned out that in this project initially behavioral requirements were totally missing, while in the project at ESOC no quality requirements were given. Although in the railway project both scenarios and required functions were initially described, the purposes of some of these scenarios were left open and had to be asked for. The lesson learned is that people often do not provide information on their own that they feel is "obvious". Without the use of our metamodel important information for the developers would have been lost.

In addition, we would like to point out the usefulness of domain-specific requirements *classes*, and the use of an *inheritance* mechanism within the corresponding taxonomy. Especially in the railway project, building domain-specific classes / subclasses of requirements — and in particular also of scenarios — was of great utility. Entering the requirements in a taxonomy helped to determine those closely related to each other. This led to the discovery of *conflicting* requirements and an explicit representation of a corresponding association between the requirements instances representing them. After the right ones were determined, the conflicting ones were explicitly labeled as *invalid* in their attribute Status. This improved the *consistency* of the requirements.

Moreover, this classification of domain-specific requirements led to the discovery of *similar* requirements and an explicit representation of a corresponding association between the requirements instances representing them. This explicitly points to *redundancy* in the definition of the requirements.

In the mission planning project for ESOC, our most valuable experience was to uncover another important relationship between requirements: the embedding of the majority of functional requirements into an overall scenario. The representation of requirements as objects has provided the whole object-oriented modeling formalism for representing all these relationships explicitly.

While in the project at CERN some initial domain model was already given, both in the ESOC and the railway projects no such information was available. Our approach of concurrently developing a domain model and dealing with the requirements proved useful there. Since these domains are quite complicated, an explicit representation of the links between the requirements and the domain model is particularly useful for having the developers understand what the requirements are talking about. Just as an example, the notion of a "window" (a time frame for accessibility of a satellite) is very different in the space domain from the every-day notion of a window or the one used in computer science.

Apart from the representational issues, we used two different *processes* of requirements specification. For the first, documents in natural language were given, that were analyzed by this author. From this source of information (enhanced by email communication), a broad basis of the representation in RETH was built. In the railway project, the natural language texts of requirements statements were distributed in several places, e.g., documents issued by ÖBB and protocols of meetings. Therefore, we also developed a model classifying the types of existing documents and their generic relationships. Since the information came from several places, it was very important to document the *source* as an attribute of the requirements objects. This greatly improved the *traceability* of the requirements.

The second process involved users and this author working together in front of the machine. Both processes had their respective advantages. The first one allowed to quickly process a large amount of information with only little communication demands. The second process allowed us to work out more subtle issues, which required much and direct communication. The representation in RETH as represented on the screen provided an excellent communication medium. From our experience, these processes can be successfully performed alternately. Their advantages are complementary. Especially the second form of process should also be looked at from the viewpoint of CSCW (Computer-Supported Cooperative Work), and a challenge would be to support it when the participants are at different places. As a first step in this direction, we built an interface to WWW (World-Wide Web) that lets people involved browse the hypertext base of RETH from distant places concurrently. This feature is especially important for CERN, since people involved are typically spread around the world.

5 RELATED WORK

Due to lack of space we cannot give here a comprehensive overview of all the proposed approaches to requirements engineering. Especially for the traditional ones, the interested reader is referred to [2]. Recent OOA approaches challenge the traditional ones [3] but still ignore early development phases where important clarifications have to be made.

In particular, the methods and tools *Objectory* [6] and KBRA [1] bear some similarity to our approach. However, they both lack important features in contrast to RETH. For instance, they both do not make a clear distinction between behavior and function.

6 CONCLUSION

There are many important advantages of using *hypertext* for requirements engineering. Generally, hypertext *links* lets users and analysts make relationships and dependencies explicit and promotes awareness of them. Moreover, these links allow the user to navigate through the representation. These more or less obvious features are of course utilized in RETH, since our approach primarily strives for being useful in practice. Additionally, we introduce the use of hypertext for requirements engineering in new ways: *incremental formalization* in a mediating representation between the completely informal ideas of the user in the very beginning and the more formal representation of domain models and requirements; a smooth integration of natural language texts in evolving object-oriented models; and support for the cognitive processes involved in modeling by a suitable external representation.

As a consequence of our experience from applying RETH, we propose to use hypertext for requirements specification. Since other approaches like using diagrams are complementary to this, we propose to combine them.

Requirements specification involves much communication between humans. When hypertext is used as an external representation supporting the activity of domain modeling, it becomes a process where human and computer co-operate. Since hypertext is a semi-formal representation, it can help bridge the formality / informality gap between computer and human. In summary, RETH provides both for a mediating representation during *incremental formalization*, and for convenient interaction of the requirements engineer with a computer.

REFERENCES

1. A. J. Czuchry and D. R. Harris. KBRA: a new paradigm for requirements engineering. *IEEE Expert*, pages 21–35, Winter 1988.
2. A. M. Davis. *Software Requirements: Objects, Functions, and States.* Prentice Hall, Englewood Cliffs, NJ, 1993.
3. D. de Champeaux and P. Faure. A comparative study of object-oriented analysis methods. *Journal of Object-Oriented Programming*, pages 21–33, March/April 1992.
4. R. Guindon. Requirements and design of design vision, an object-oriented graphical interface to an intelligent software design assistant. In *Proc. CHI '92*, pages 499–506, Monterey, CA, May 1992.
5. F. Halasz and M. Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, February 1994.
6. I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. *Object-Oriented Software Engineering: A Use Case Driven Approach.* Addison-Wesley, Reading, MA, 1992.
7. H. Kaindl and M. Snaprud. Hypertext and structured object representation: A unifying view. In *Proc. Hypertext'91*, pages 345–358, San Antonio, TX, December 1991.
8. F. M. Shipman and R. McCall. Supporting knowledge-base evolution with incremental formalization. In *Proc. CHI '94*, pages 285–291, Boston, MA, April 1994.
9. M. Snaprud and H. Kaindl. Types and inheritance in hypertext. *International Journal of Human-Computer Studies*, 41(1/2):223–241, July/August 1994.