Model-Based Requirements Engineering with AutoRAID

Bernhard Schätz, Andreas Fleischmann, Eva Geisberger, Markus Pister

Fakultät für Informatik, Technische Universität München
Boltzmannstr. 3, 85748 Garching, Germany
{fleischa|geisberg|pister|schaetz}@in.tum.de

Abstract: While software design is increasingly based on models, requirements engineering is generally performed using structured text; as a consequence, only a weakly structured connection to design is possible. Model-based requirements engineering can bridge this gap. With AutoRAID we introduce a tool prototype supporting a structured and integrated requirements model, and operations for refinement and structuring leading from textual requirements to the design model.

1 Introduction

The core issue of model based requirements engineering is the introduction of a *common conceptual model* for requirements *and* design elements combined with step-wise *formalization of requirements* in the development process. Such a gradual, tool-based transition from requirements analysis to design reduces the gap between these phases. We introduce the tool AutoRAID for that transition, based on a review-like process.

Due to their generally informal nature, ensuring the quality of requirements demands different techniques then applied when assuring the quality of design specifications. Standards like IEEE 1233-1998 [IE98a] define criteria for requirements specifications to ensure quality in the early phases of development. Here, we address five quality defects by means of constructive and analytic quality assurance:

- **Inconsistency**: By breaking down requirements into a detailed model with linked elements, inconsistencies are removed (e.g., when detailing scenarios of a system based on its interface).
- **Ambiguity**: By using a detailed model for the requirements, expressing the requirements in terms of this model successively leads to elimination of ambiguities (e.g., when classifying and detailing the steps of a scenario).
- **Intraceability**: By linking the detailed requirements on the fly to elements of the design, traceability is automatically ensured (e.g., when motivating a mode of the system out of a mode constraint).
- **Infeasibility**, **Verifiability**: Through the restriction imposed by formalization, a design-like precision of description is enforced, reducing the risk of infeasibility and verifiability (e.g., when specifying the interactions performed during a scenario).

While generally manually performed, review- or form-based approaches are used; here we present a model-based and tool-supported approach, based on a review-like process. Model-based requirements engineering uses a formal model of requirements focusing on their classification, implicitly present in approaches [IE98b]. Thus, it fits into a struc-

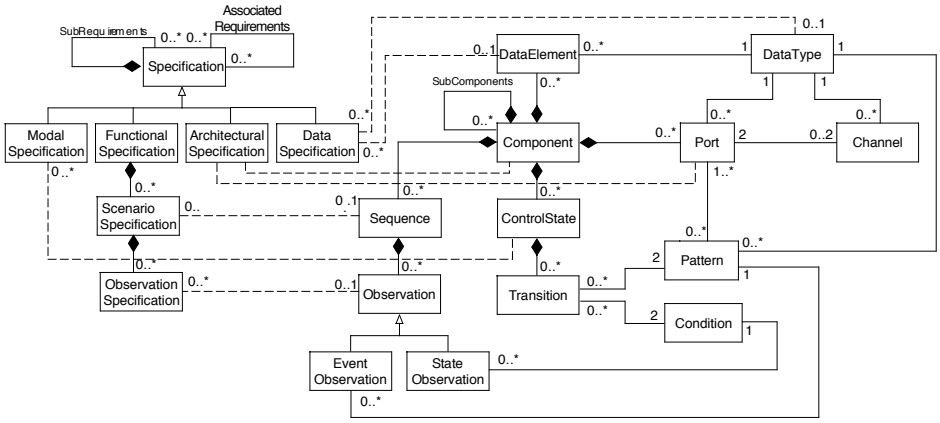

Figure 1: Simplified Conceptual Model for Analysis and Design

tured development process when tool-supported structuring and classification mechanisms as used in a review process [DP04] are offered to domain experts. Besides those structuring steps (e.g., identifying the steps of a use case), analytical techniques applied in an inspection are mechanized (e.g., each functionality must allocated to a component). In the following, we present those techniques as implemented in AutoRAID [AR04]:

- *Structured requirements model:* Introducing domain-specific classes of requirements (e.g. architectural requirements, modal requirements, use case scenarios).
- *Integration of textual requirements:* Integrating textual requirements into the model and offering pragmatic mechanisms for their use.
- *Tool-based support for classifying and structuring*: Mechanizing the step-wise structuring (e.g. identification of actuators in functional requirements).
- *Connection to design-oriented model elements:* Linking classes of requirements and classes of design (e.g. modal requirements to control states).

The model of design is taken from the AutoFOCUS [Sch04], targeting reactive systems.

2 The AutoRAID Conceptual Model

State-of-the-art requirements tools generally use a rather simple conceptual model to define requirements specifications, basically supporting only the concept `Requirement' as well as the relations `sub-requirement' and `associated requirement'. In contrast – as shown in a simplified version in Figure 1 – AutoRAID uses a more complex model, built around the notion of a (informal) specification element. As explained in Section 3, our approach consists of the activities *Structuring*, *Classification*, *Formalizing* and *Analysis*. To support them, the model offers specific elements and associations described below.

2.1 Structured Requirements

A major step during requirements engineering – as found in DOORS or Requiste Pro – consists of breaking down requirements into sub-requirements. AutoRAID uses the con-

cept *Specification* (corresponding to a single requirement) and associations *SubRequirements* and *AssociatedRequirements*. Specification elements have the usually attributes (e.g., unique identifier, creator, priority). Requirements can be broken up into sub-requirements forming a hierarchy that allows back-tracing the requirements to their goals.

2.2 Classified Requirements

As suggested in [IE98b], specifications can be classified as:

- *Architectural requirements*, describing the structure of a system. Components, interfaces and channels are captured with these requirements.
- *Modal requirements,* describing the operating modes of the application. Different states or modes of the system and its components are defined here.
- *Data type requirements*, describing data types that can be used for communication within the system and with the environment.
- *Functional Requirements*, defined in terms of uses cases described by representative sequences of observations about the system and its interactions with its environment.

2.3 Integrated Requirements

Since a major functionality of AutoRAID is the easy transition from an informal to model-based description of the system, the model shown in Figure 1 shows a tight integration of concepts for both kinds. For each classified informal concept (e.g*., architectural specification*) there is a model-based concept (e.g*., component, channel, port*) associated with it. The dashed associations show the connections between the respective concepts. These links from the requirements to a structured models support analysis and completion of the specification as well as traceability between analysis and design.

3 Tool Support

Figure 2 shows the user interface of AutoRAID. The model browser (left hand side of Figure 2) shows the tree of the requirements analysis (*Analysis*) as well as of the design-oriented model (*Component Car*); it shows generic *Specifications* (*Requirements*) as well as classified *Specifications* (*Use Cases, Constraints*). The work area (right hand side) shows the formalization step (see Subsection 3.3) of an *Event Observation* (front layer); this communication event itself is a part of a *Scenario Specification* (second layer), which in turn is a part of a *Functional Specification* (background layer). AutoRAID assumes a review-based development process, with the steps defined in the following subsection.

3.1 Generation and Import of Requirements

The tool AutoRAID allows to manually create or import requirements. During creation, a requirement is automatically assigned a unique ID; the user adds attributes like *title, status, priority,* etc. When creating requirements from an imported document, a `select-and-create' functionality allows to conveniently define a requirement. To support trac-

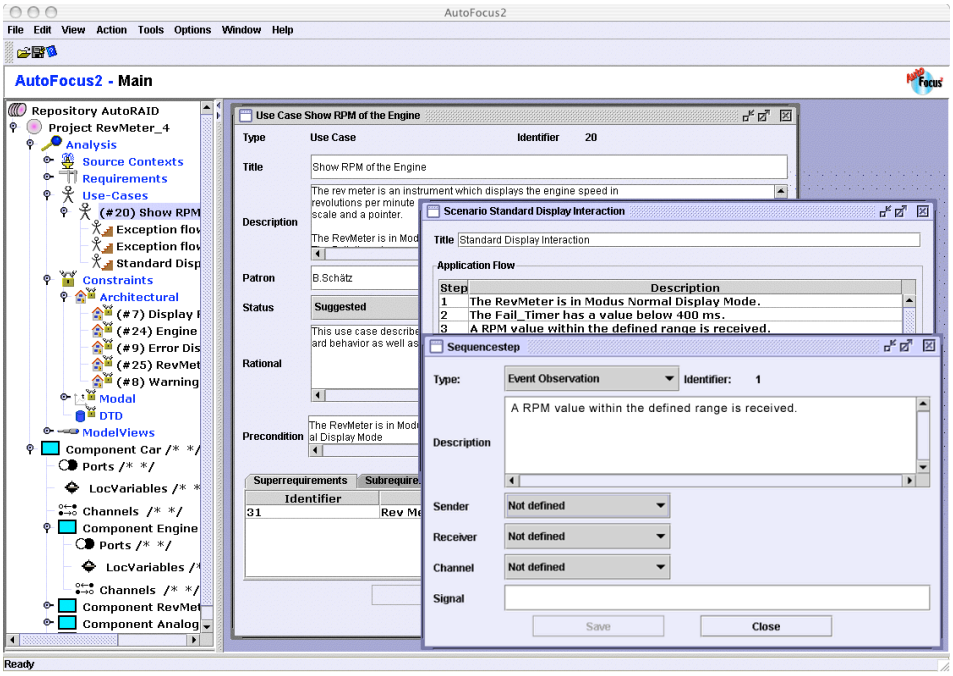

Figure 2: Definition of Event Observation in Scenario “Standard Display Interaction”

ing, the link between text and requirement displayed in the source text. For documentation purposes, a structured document can be generated from the refined requirements.

3.2 Structuring Requirements

To support a hierarchical structuring of a requirement specification, any kind of *specification* can be structured hierarchically. The possibilities in AutoRAID are:

- A *Specification* can be linked to one or more *Specifications,* to identify common *SubRequirements* or correlated requirements.
- Out of an existing *Specification* a new *Specification* is created and linked to it as a *SubRequirement*, to break up a compound specification into its sub-specifications.

AutoRAID offers a convenient `select-and-create’ functionality, to easily identify a part of a *Specification* to become a separate sub-specification; this functionality reuses the selected part of the original description to create the new *Specification* and establishes a *SubRequirement* relation between them.

3.3 Classifying and Formalizing Requirements

By classification, generic requirements are grouped according their category; specifications are broken down into sub-specifications until they are detailed enough to fit one of the categories (*architectural, modal, data, functional*). To enforce more preciseness, categorized specifications are further detailed. E.g., a *Functional Specification* can be

structured in form of a *Scenario* identifying its *Observation Specification* steps. These steps are again classified, e.g., as communication *Event Observations* with sender, receiver, and signal. Using a form-like, text-based description, at this level a specification is already formalized, requiring the strictness and preciseness of a detailed conceptual model. These specification elements can be displayed in a graphical fashion; e.g., a scenario is alternatively displayed as a sequence diagram. Thus, AutoRAID allows giving a detailed description of the system from the requirements point of view, adding, e.g., rationales to document development decisions; or, from the design point of view, focusing on a compact and comprehensible description. By using shared elements, design elements can be easily traced back to motivating elements of the requirements model.

3.4 Analyzing Requirements

AutoRAID supports the step-wise analysis in a review-based and constructive fashion using the strictness of model-based formalization step implicitly to analyze a specification. Additionally, using the structured, model-based specification incrementally developed from weakly structured specifications, analysis techniques in form of consistency conditions can be applied to detected possible weaknesses of the model, e.g., "Each business requirement is refined by at least on application requirement" or "Each application requirement is classified or refined by a application requirement". Consistency analysis is performed automatically, presenting those specification and model elements that do not meet the consistency conditions.

4 Conclusion

AutoRAID covers structuring, classification and analysis of requirements, capturing the decision process through forward and backward tracing, supporting the incremental transformation of an informal specification to a model-based design. Current research issues address the inclusion of additional classifications (e.g., work flows) and the treatment of product lines. Furthermore, the applicability of the AutoRAID approach is investigated in real-world case studies in the automotive (chassis electronics) and the aviation domain (flight control).

5 Bibliography

[AR04] Website of AutoRAID, http://wwwbroy.in.tum.de/~autoraid/

[DP04] Denger, C. Paech, B. "An Integrated Quality Assurance Approach for Use Case Based Requirements". In: Rumpe, B. Hesse, W. (eds). Modellierung 2004. Springer, 2004.

[IE98a] Software Engineering Committee of the IEEE Computer Society. IEEE Guide for Developing System Requirements Specifications. IEEE Standard 1233-1998. IEEE CS.

[IE98b] Software Engineering Committee of the IEEE Computer Society. IEEE Guide for Software Requirements Specifications (ANSI). IEEE Standard 830-1998. IEEE CS

[Sch04] Schätz, B.: Mastering the Complexity of Embedded Systems. In: Formal Techniques for Embedded Distributed Systems, Kordon F., Lemoine, M. (eds.). Kluwer, 2004.