Using RE Knowledge to Assist Automatically during Requirement Specification

Thorsten Merten, Thorsten Schäfer, Simone Bürsner
Department of Computer Science
Bonn-Rhine-Sieg University of Applied Sciences
Sankt Augustin, Germany
{thorsten.merten,simone.buersner}@h-brs.de
thorsten.schaefer@smail.inf.h-brs.de

Abstract—In a two semester software engineering (SE) course at Bonn-Rhine-Sieg University students have the opportunity to actually elicit, analyze and document requirements as well as design and develop a correspondent software product in teams of approximately four. The students have to use an issue tracking software in combination with a Requirements Engineering (RE) tool to document and plan their work. Though the course starts with RE theory from elicitation via documentation and traceability, we found that the students find it difficult to combine different RE artifact types and to develop useful traces between them. In this paper we present an approach to provide feedback and give pro-active advice inside an RE tool, while the specification is created1. To derive this feedback we use a knowledge base containing rules and best practices to create a requirements specification. An assistance system applies these rules to guide the user in different situations, beginning with an empty specification up to the implementation of various RE artifact types and traces between them.

This paper presents the status of our knowledge-based feedback mechanism and possible extensions. In order to get primary indicators for the value of this approach we did experiments and workshops with eight students who worked with the same tool with and without the feedback system.

Keywords—RE; knowledge engineering; teaching; direct feedback, software-based feedback agents

I. INTRODUCTION

A. Motivation

This paper is motivated by a beginners software engineering (SE) course, where students have the opportunity to develop a software product from scratch by accomplishing all the necessary software engineering activities, especially Requirements Engineering (RE) work. The students are intended to use an issue tracker as well as a RE tool to document their works. We noticed a lack of structural thinking when the students build their RE documentation, especially when different RE artifact types had to be combined (e.g. stakeholders, goals and scenarios). Even though the concepts behind RE artifacts and traceability as well as the modelling tool are taught in depth, the students have difficulties combining multiple artifact types to a single specification and traces are seldom built. Often, it is not possible to give feedback directly such that the problems persist until the next reviewed milestone or in the worst case until the students start implementing the software. Making use of a variety of RE artifact types and their combination is an important skill, because “to construct a reasonable picture of reality, you need to put together interlocking models [and] many interwoven requirement elements [1, p. 376].”

The approach presented in this paper deals with the above issues using direct and unobtrusive feedback on the RE specification’s status by the means of knowledge engineering. In the next section we present related literature and tools using similar techniques. In section II our approach is presented. As opposed to other feedback approaches our tool supports multiple RE artifact types and the documentation of traces amongst them.

B. Software-Based Feedback Agents

Different approaches use software-based feedback agents (SBFA) to guide the user in certain tasks. SFBAs have the ability to access some kind of knowledge or best practices. They then analyze the input data and compare it to the knowledge repository to give direct feedback to the user of the software. For teaching purposes this gives some opportunities, e.g.

- to give feedback without the need of human resources,
- to give advice regarding the further elaboration and
- to give feedback immediately (before an expert reviews the assignments).

The last two aspects distinguish SFBAs from other (non-immediate) feedback mechanism as for example consistency checkers or orphan detection, since they act neither preventative nor immediately.

Commercial RE tools so far do not provide SFBAs. They generally offer static or dynamic help systems or help tutorials (see Table I). Those systems do not offer a methodical guidance to the user, they only explain how to use software itself. However, scientific approaches use SFBAs for different tasks in RE. Gleich et al. [2] present an approach that processes well formed natural language requirements and gives hints about ambiguity problems within these requirements. Knauss et al. [3] present a rule-based-reasoning approach on use cases and textual requirement de-
Online Help and Feedback Systems

<table>
<thead>
<tr>
<th>Software</th>
<th>Help system</th>
<th>SBFA</th>
<th>SBFA technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOORS</td>
<td>help</td>
<td>no</td>
<td>n/a</td>
</tr>
<tr>
<td>ReqMan</td>
<td>help &amp; video tutorial</td>
<td>no</td>
<td>n/a</td>
</tr>
<tr>
<td>HeRa [3]</td>
<td>no</td>
<td>textual requirements &amp; use cases &amp; glossary</td>
<td>rule-based reasoning</td>
</tr>
<tr>
<td>L’ECRITOIRE [4]</td>
<td>n/a²</td>
<td>scenarios</td>
<td>text mining</td>
</tr>
<tr>
<td>RE Checker [2]</td>
<td>no</td>
<td>nat. language requirements</td>
<td>text mining</td>
</tr>
<tr>
<td>This approach</td>
<td>yes</td>
<td>multiple artifact types and traces</td>
<td>rule-based reasoning</td>
</tr>
</tbody>
</table>

Table I

In our approach we established a web-based SBFA system that uses reasoning to assist the user through three types of direct feedback:

1) To create single requirement artifact types (e.g., you did not enter required data for field x)
2) To build a structure of requirements (e.g., you may want to start your project by defining a project vision)
3) To assist with horizontal traceability between multiple requirement artifacts (e.g., the use case you defined is associated to neither any actor nor a goal)

The value of automatic assistance for properties or text-fields through text-mining within certain artifact types (1) has already been shown ([3], [6] and [7]). However, especially in teaching it is important for the students to be able to experience the whole specification life cycle. The focus of this work is to provide assistance on the structure of a requirements specification (2) as well as to establish traceability between requirement artifacts (3).

A. Rule Derivation

For the rule derivation we follow a classic knowledge engineering approach. Firstly, knowledge is acquired, e.g., we describe a domain model and define a specification procedure, which we also introduce to our students, mainly from [1], [8] and [9]. Within this domain model, different types of rules are created that correspond to the types of feedback described above.

1) Rules for single requirement artifact attributes. Such as, if a technical solution is given, it should mention the responsible stakeholder or system [9, p. 38].
2) Procedure rules. These rules describe semantical dependencies between certain requirement artifacts.
3) Artifact-type-interdependent rules. These rules describe candidate relationships for horizontal traceability.

Each rule then gets categorized as artifact specific or procedure specific. The rules are tagged according to their RE artifact type and coded as Ruleby⁴ to connect them to our web-based RE tool.

Rules in category (2) define logical implications when creating an empty specification. This said we do not model an actual process since RE should be an incremental and continuous activity [8, p. 48]. As for example, rules of the following format are defined: After creating a RE artifact of type project vision a set of artifacts of type goal should be defined. Once students understood these concepts and created their first artifacts the rules are not applied to further goals. Therefore they are free to find their own process. Our SBFA is generally designed to be unobtrusive. To stick to the outlined example, the user is able to follow a different procedure than the SBFA proposes. She will not be hindered in modeling requirements by the SBFA. Only advice for the

²Table I shows a selection of commercial tools. The set of help features is representative for ten commercial tools we analyzed, as well as the scientific approaches presented above.

³not mentioned in the paper

⁴https://github.com/codalytics/ruleby
overall specification or the current state of the project will be given.

B. Reasoning

To test this approach we implemented rules to systematically build a specification under consideration of the following RE artifact types:

1) Project vision: A high level description of the project, often referred to as the elevator pitch.
2) Goal: Business and stakeholder goals that should be derived from the project vision.
3) User profile: A description of a stakeholder (group), e.g. a persona [10].
4) Work area: A structural element to group several coherent use cases or user tasks.
5) Scenario: A single scenario, e.g. a user story [10].
6) Task: A happy day scenario description similar to use cases with several additional developments. Especially a solution oriented column [9].

The user of our tool is guided through the creation of a requirements specification. The SBFA acts unobtrusively, so the user can specify arbitrary artifacts ignoring its guidance. In the latter case the system gives advice on the structure and horizontal traceability for the artifacts that have already been specified.

The SBFA uses a rule-based inference mechanism. The rules can be divided in three categories:

- **RA**: Describing properties of requirement artifact types
- **RP**: Describing states of the standard procedure we propose for the particular SE course
- **RT**: Describing edges in the traceability graph

Furthermore the rules are applied in two different ways.

1) **General Reasoning**: The inferred information from rules in the category **RP** as well as a subset of **RT** are presented to the user in overview modes of the tool. An example is the front page of the requirements editor. Figure 1 shows the inference of the **RP** rule: IF a project vision exists but no project goal exists THEN give an advice.

2) **Context Aware Reasoning**: The assistant processes rules that refer to a certain artifact if rules of type **RA** or **RT** do not apply to a broader context. Figure 2 shows the inference of the **RA** rule IF an example solution is given for a subtask THEN give an advice from literature for this field.

An example of a **RT** rule is shown in Figure 3, which applies to the constraint IF a task has been defined but this task is not connected to a user profile THEN give an advice.

C. Architecture and Rule Scope

Since the underlying RE tool distinguishes different artifact and different trace types, traceability rules (**RT**) as well as procedure rules (**RP**) can be applied precisely by processing the traceability graph and its nodes. The structure of the requirements specification is also created with traces (which we call parent/child traces). Therefore the SBFA can use **RT** and **RP** rules to assist with the specification structure as well. In the following we define meta rules, for each meta rule the knowledge base allows to create concrete rules which apply to a specific situation during the specification. At the moment more than fifty concrete rules are available.

In the following two lists we describe the scope of the SBFA using meta rules and give corresponding examples of concrete rules as well as the according literature we used to create the concrete rules.

- **M1)** Missing trace (of a certain type) from or to another RE artifact. **M1-Example 1**: If a project goal is not derived from a project vision tell the user to create a trace (derived from lecture notes). **M1-Example 2**: A task is not connected to a user profile or stakeholder tell the user to create a trace [11, p. 34].
- **M2)** Wrongly created trace from another requirements artifact. **M2-Example**: If a task is organized in a section instead
of a work area tell the user to move the task to a work area (and create a work area if there is none) [9, p. 34]

$M_4$ Requirements artifact is not organized within a section. $M_3$-Example: If an artifact of type $t$ with $t \neq$ section or task is defined in the root project instead of a section tell the user to move it into a section.

$M_5$ Requirements artifact of a certain type is not organized within the according structuring artifact. $M_4$-Example: If a task artifact is not created below a work area tell the user to move it accordingly [9, p. 34].

$M_6$ A set of requirement artifact types $A$ exist, but another artifact type which should be created (next) is missing. $M_5$-Example 1: If a set of stakeholders and a project vision exist but the project has no goals tell the user to specify goals for that vision [1, p. 51, p. 64] $M_5$-Example 2: If $A$ is the empty set tell the user to start with a project vision [1, p. 28].

Furthermore the tool provides requirements templates, such that we do not need to process completely unconstrained text to apply rules of type $(R_A)$. Requirements templates allow us to check for the following types of meta-rules:

$M_7$ General advice for specifying an artifact. $M_8$-Example: If a project vision is specified tell the user how to formulate the vision [1, p. 53]

$M_7$-Hints for certain fields of the template if they are used. $M_7$-Example 1: If a task specification uses the template field difficulties tell the user to use the field sparingly and if possible reformulate as problem using subtasks [9, p. 36]

$M_7$-Example 2: If a subtask specification uses the template field 'solution' tell the user that specified solution should be non trivial [9, p. 38]

$M_8$ Warnings for 'important' empty fields. $M_8$-Example: If the description for a work area is empty tell the user to give a brief summary about the purpose of the work area [12]

$M_9$ Warning for (missing) keywords in certain fields. $M_9$-Examples: If the task or subtask descriptions do not use imperative language tell the user to rethink it [11, p. 36]

As it can be seen from the meta-rules the overall architecture allows to specify rules for each requirements artifact as well as each trace type. It is extensible towards other reasoning mechanisms than inference based reasoning as well as other techniques like text-mining to deeply inspect certain attributes of RE artifacts.

The SBFA displays three outputs for each rule. Firstly it displays the derived advice. An advice is displayed with one of the three severity levels, namely 'hint', 'warning' or 'problem'. The severity has to be decided by the rule author. The second output is a link named 'Learn why this is important' which reveals a short text (see Figure 2). The text is a description we extracted from the literature which explains why one should follow this advice. The third part, if applicable, is a link named 'Click here to fix this' which points the user to the next action that should be taken. The action differs according to the given advice. If an artifact is missing the link points to the editor for the according artifact type, if a trace is missing the link points to the traceability editor and so on. This link is displayed only if $a$) the system can reliably determine the next action and $b$) the next action does not automatically change the data.

Below an example is given of the complete input, processing and output chain of a $R_T$ rule:

**input** Traceability graph.

**check** Is there any artifact of type 'scenario' which is not traced to an artifact of type 'goal'.

**advice** This scenario does not belong to a goal.

**severity** Warning.

**learn** You should connect this scenario to a goal because scenarios describe the fulfillment of a goal. This helps you to justify and classify your scenario. If there is no appropriate goal you should think about creating a new one.

**link** Point to the traceability editor unless there are no artifacts of type goal. In this case point to the goal editor.

### III. Conclusion

We tested the concept with a group of eight students. They worked with the RE tool with and without the SBFA enabled. In a qualitative questionnaire we asked them to give feedback and explain their experience with the SBFA. All of them appreciate the early feedback on their work. They used the SBFA system as well as the help system continuously during their specification work.

During a workshop, where we collected and discussed their feedback to the SBFA, they made several concrete suggestions for sustaining the assistance role during specification work. For instance, they demanded an aggregated view of all advices. They wanted more rules for an even more context sensitive support and rules to alert if something has already been done wrongly. Right now we focused on guiding the user. They asked for more features in order to obtain an assistant that guides continuously in the process of creating and completing the specification and resolving defects. Their vision went far beyond initial support for inexperienced users to structure their requirements specification and combine multiple artifact types as the students compared the role of the SBFA to a dynamic and adaptive context specific tutorial. For example they demanded that the 'learn why this is important'-link is connected to a community maintained knowledge base which can be continuously improved and enables the discussion of certain rules. Although the students described the SBFA system as a valuable kind of “walk-through assistance” and appreciated the guidance to develop artifacts in a structured way as well as to understand the relations between artifacts, they asked for a feature to disable certain (categories) of advice once they were understood, which is a common demand when working with SBFAs but has not been implemented yet. The
most appreciated feature surprisingly was the 'click here to fix this'-link. The students noticed that this link helped them to create a specification very quickly without investing too much gumption. The combination of both, an increased workflow and the possibility to learn more about the next step was what spurred them on. As we said before our system cannot improve the quality of texts within artifacts as it would be necessary for example to assist with the abstraction level of use case descriptions since we do not use any text mining or text processing besides a keyword search. Therefore the details and implementation of the artifacts themselves have to be checked manually by the course instructor. However, the system is able to give advice on the specifications’ structure, which is important to provide the ‘big picture’ of a specified system.

SBFA systems can be noticed as ‘annoying’ [5] and we therefore need to improve several aspects of the system as for example the ability to let the user hide an advice. Furthermore, to use the same system in industrial settings, rules have to be editable by team members and rationales for rules need to be transparent. However, the system can be adopted to different project settings and reflect documentation regulations making it possible to use it in commercial settings as well.

Although we used an open source tool to implement the proposed concepts, most commercial tools provide extension points, plug-in concepts or scripting languages through which SBFA could be implemented. One example is the DOORS extension language (DXL).

IV. Future Work

The approach will be validated in several case studies in undergraduate courses. We plan to use multi-variant tests to improve the SBFA system itself and most importantly find empirical evidence that the number of errors we encounter during manual correction can be lowered by this approach. Furthermore it is planned to extend the approach to support more artifacts types as well as diagrams and diagram creation. To achieve this, we need to develop and derive further rules. So far two strategies seem to be applicable:

1) Rule sets: A rule set represents a certain approach or project setting (e.g. Cockburn’s use cases [13] or Cohn’s user stories [10]). A rule set can be loaded in the requirements editor and helps for this certain approach.

2) Concept rules: Concept rules combine different artifacts types (e.g. use cases, task descriptions and user stories). They shall describe commonalities and differences within certain RE artifact types. This renders possible to categorize exchangeable artifact types. E.g. both use cases and user stories provide scenario descriptions.

Implementing those extensions enables the system to support various RE techniques and most importantly their combinations. Overall the solution is a promising approach to support teaching. Furthermore the rule-based system can be adapted to facilitate documentation constraints in enterprises.

ACKNOWLEDGMENT

This work is partly funded by the BMBF (German Federal Ministry of Education and Research) within the research project KoREM (http://www.korem.de).

REFERENCES


