Empir Software Eng (2006) 11: 303–329
DOI 10.1007/s10664-006-6405-5

An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development

Johan Natt och Dag · Thomas Thelin · Björn Regnell

Editor: Daniel M. Berry

Abstract This paper presents an experiment with a linguistic support tool for consolidation of requirements sets. The experiment is designed based on the requirements management process at a large market-driven software development company that develops generic solutions to satisfy many different customers. New requirements and requests for information are continuously issued, which must be analyzed and responded to. The new requirements should first be consolidated with the old to avoid reanalysis of previously elicited requirements and to complement existing requirements with new information. In the presented experiment, a new open-source tool is evaluated in a laboratory setting. The tool uses linguistic engineering techniques to calculate similarities between requirements and presents a ranked list of suggested similar requirements, between which links may be assigned. It is hypothesized that the proposed technique for finding and linking similar requirements makes the consolidation more efficient. The results show that subjects that are given the support provided by the tool are significantly more efficient and more correct in consolidating two requirements sets, than are subjects that do not get the support. The results suggest that the proposed techniques may give valuable support and save time in an industrial requirements consolidation process.

Keywords Requirements management · Software product development · Linguistic engineering · Natural language requirements

1. Introduction

Requirements engineering is generally regarded as an important success factor in software development (Hofmann and Lehner, 2001). One of the activities within

J. Natt och Dag (✉)
Obigo AB, Scheelevägen 17, SE-223 70 Lund, Sweden
e-mail: johan.nattochdag@obigo.com

T. Thelin · B. Regnell
Lund University, Lund, Sweden

requirements engineering, requirements management, deals with requirements storage, change management, and traceability issues (Sommerville, 2001). These activities are generally challenging. In market- and technology-driven software development they are exceptionally difficult due to the particular characteristics of the development situation (Lubars et al., 1993; Novorita and Grube, 1996; Potts, 1995; Regnell et al., 1998; Sawyer et al., 1999; Yeh, 1992). Fundamental organizational issues, such as the primary goal, the success measurements, and the product life cycle, present challenges not found in bespoke, or contractual, development. Unfortunately, many industrial requirements management activities are left untouched by the research world and there is an urgent need for a better understanding and improved support (Brinkkemper, 2004).

This paper presents an experiment with a new open-source tool, which incorporates linguistic engineering techniques to support a specific requirements management activity: consolidation of large amounts of requirements that are elicited either from different stakeholders or from the same stakeholder at different points of time. The general purpose of the consolidation process is to find the overlap of two requirement sets with respect to the functionality they convey.

The experiment was designed to capture the consolidation process in an industrial case, where requirements consolidation is particularly challenging and where much work could be saved if consolidation was better supported. The purpose has been to investigate if the proposed tool and the underlying techniques may give adequate support in that process. The two subjects group were asked to consolidate two requirements sets by finding and linking requirements that address the same underlying functionality. The groups were given two different version of the support tool. One version comprised linguistic engineering techniques aimed at giving automatic assistance in finding similar requirements. The other version offered the possibility to submit search terms for finding similar requirements.

The main results show that the subjects in the group using the tool with automatic assistance performs, on average, significantly better. Not only are they able to consolidate requirements faster, but they are also doing it more correctly—assigning more correct links and missing less.

The paper is structured as follows. In the next section, the general background to the consolidation process is given. This is followed by a problem description captured from our industrial partner. The experimental conception, preparation, planning, and operation are then presented in Sections 4 through 7. The analysis of the experimental results are presented in Section 8. A short discussion is found in Section 9 followed by Section 10, which summarizes and concludes the paper.

2. Background

Market- and technology-driven software development companies essentially concentrate on three parameters in order to succeed (Novorita and Grube, 1996; Sawyer et al., 1999):

- The point of time when new releases shall reach the market, i.e., how well the market window is targeted.
- The release content, i.e., the quality and functionality of the product release.
- The cost of development, comprising the full lifecycle of the release.

To increase market share and decrease risks, companies concentrate on serving many needs within one market [i.e., *market specialization* (Kotler, 2002)]. To fulfill specific needs within the market, customization and customerization has been adopted. *Customization* allows end-users to buy individually differentiated products within certain prescribed aspects (e.g., by individually specify the components to be included in a new computer). *Customerization* goes one step further and involves the interaction with each customer, allowing each customer to request customization on a certain product.

To optimally enable market specialization and product customization in software development, software engineering has embraced the concept of product lines (Clements and Northrop, 2002). A *software product line* is a family of software-intense products that share a common set of features. Each individual product is then developed from a common set of assets in a controlled manner. The objective of a software product line is to optimize software engineering effectiveness and efficiency and to gain improvements in productivity, cost, and quality (Krueger, 2003; Clements and Northrop, 2002).

For software product line development in a market-driven company, the requirements engineering activities differ compared to those for single systems in the following ways (Clements and Northrop, 2002; Potts, 1995):

Requirements elicitation. A larger set of stakeholders, such as marketing, support, development, testing, usability evaluations, and technology forecasting. To remain competitive on the market, requirements collection is insufficient and new requirements must also be invented within the organization based on foreseen end-user need. For individual customer satisfaction, possible and anticipated variations points must explicitly be captured.

Requirements analysis. A focus on identifying commonalities and variations in order to identify reuse. Close contact with potential customers enables feedback and negotiation of opportunities for saving money by using more common features in favor of unique ones.

Requirements specification. Two levels: product-line-wide requirements and product-specific requirements. The product-line-wide specification must indicate where variations points are possible. The product-specific requirements complete or extends the product-wide specification.

Requirements verification. Occurs in two stages according to the two levels of specification.

Requirements management. Must acknowledge the two levels of specification and the two stages of verification. Impact of changes to the product line must be systematically assessed with respect to all the products in the product line. Links between the core assets and the product line requirements must be identified and maintained.

These activities are conducted iterativley and in parallel throughout the development process, making it increasingly important to maintain an effective and efficient requirements engineering process. When a company that has embraced the product line approach is growing, more product variations and customization will be part of the product line. Each individual product has its own lifecycle and so does the whole product line. Thus, new product lines may be started in parallel with already existing, reusing parts of the soon to be phased-out product line.

To effectively manage commonalities and variations points, requirements are stored in a central repository. Changes are made to the requirements in the repository and project requirements specification are extracted on a needs basis. With the possibility to assign links between requirements to indicate interrelationships, e.g., commonalities, requirements from different customers may be kept separated. This enables better communication with each customer.

New requests arriving from customers and new requirements invented within the organization must therefore be *consolidated* with the requirements already in the repository. Generally speaking, the consolidation process is the process of finding the differences between the requirements in the repository and the newly arrived requirements, in terms of new and changed requirements.

When the product line approach is scaled up, the number of stakeholders increases and the complexity of both the product and development escalates. As a consequence, requirements arrive at a higher pace and in larger volumes, which require more time for the identification of commonalities between stakeholders. Meanwhile, the time-to-market constraint remains.

Eventually, companies face the challenge of dealing with huge information flows that may overwhelm their management and analysis capabilities. Due to the increasing difficulty of identifying and maintaining requirements interrelationships, the consolidation process becomes overloaded, resulting in requirements repository deterioration. The effects can be serious, impeding proper decision-making and customer satisfaction.

3. Industrial Problem Description

The general problem described in the previous section has been identified through collaboration with our different industrial research partners. One of our partners, Sony Ericsson Mobile Communications AB (SEMC), has found it particular important to resolve their instance of the problem. Our collaboration has comprised investigations into their situation and a preliminary evaluation of the techniques that are used in a proposed support tool (more on the tool is found in Section 5.2). This section provides a description of our industrial partner's development situation and their consolidation problem in the requirements management process.

SEMC develops mobile phones for a global market. As a market- and technology-driven company with many stakeholders, SEMC must handle requirements from a number of sources. As shown in Fig. 1, the stakeholder situation presents a more complex flow of requirements than in typical single-systems software projects.

The arrows in the figure indicate the flow of requirements. In the top left part of the figure, the hardware platform vendors are shown, represented by the two companies Ericsson Mobile Platforms and Broadcom. SEMC's technical requirements on the platform are stored in a Technical Requirements Specification (TRS). This document is managed by the Chief Technology Officer (CTO), who is also responsible for negotiation with the platform vendors.

In the right part of the figure, inputs from the market, end users, and concept studies are shown. Information from the market is gathered from sales and marketing, and through business intelligence and competitive analysis. Although SEMC does not sell their phones directly to end-users, requirements stemming from

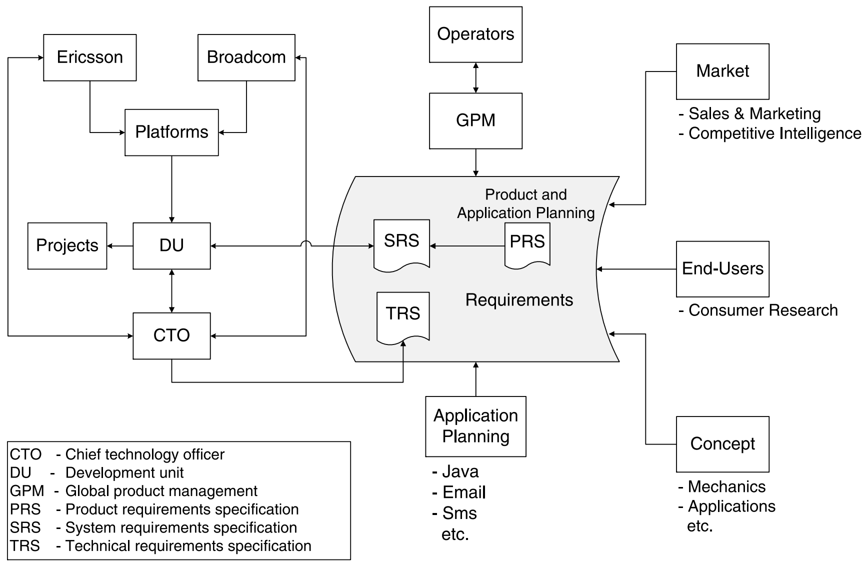


Fig. 1 Requirements sources

their own consumer research programs comprise a valuable source of information for future undertakings. Ideas that may provide competitive advantage are evaluated through concept studies and may generate requirements on both hardware and software.

Application planning, shown at the bottom of the figure, are responsible for the different application areas (e.g., Games, Java, etc.). They act as internal customers who have their own set of requirements on the phone, which must be fulfilled in order for the applications to integrate nicely in the phone.

Finally, at the top of the figure, SEMC's primary customers, the operators, are shown. The operators are responsible for selling the phones to the end user, either directly or through a third party. Through the close contact of Global Product Management (GPM) with the operators, SEMC provides mobile phone customization. In order to optimize software engineering effectiveness and efficiency, this is managed through a product line approach. Requirements are gathered and negotiated through a process that is experiencing severe overload. This process is described in Section 3.1.

A mobile phone project is initiated by a pre-study on the Product Requirements Specification (PRS), essentially comprising high-level requirements stemming from the operators. Further input in the initial phase are the concept study reports and consumer research reports. The output from the pre-study is an initial version of the System Requirements Specification (SRS), which comprises both hardware and software requirements. The responsibility for the System Requirements Specification is assigned to a development unit, DU, which are the developers of a project.

3.1. The Request for Information Process

In order for the operators to acquire knowledge in the technical capabilities of SEMC's phones, Requests for Information (RFI) are submitted to SEMC by the operators. Two kinds of RFI's can be identified: general information requests and requests for statement of compliance. Statements of compliance, which are the most common ones, comprise specific requirements and are replied upon using simple standardized statements on whether or not a stated requirement is fulfilled by the product.

The RFI process is depicted in Fig. 2. Each year, each operator submits a couple of RFIs. The RFIs arrive to the Key Account Managers, one for each major operator, in different document formats (PDF, MS Excel, MS Word, etc.) and at different times. The main specification technique for the RFI requirements is feature style (Lauesen, 2002) using natural language as the specification language.

The Key Account Manager passes the RFI on to a Bid Support Specialist, who reviews the RFI from a market point of view and decides which products shall be considered when dealing with the RFI. The Bid Support Specialist then passes the documents on to the Coordinator, who analyzes the RFI and the accompanying instruction and then distributes relevant parts of the RFI to different Areas of Expertise.

An Area of Expertise consists of a Function Group and a Technical Work Group. The Technical Work Group focuses on road maps (i.e., future functions) and the Function Group is dedicated to implementation and testing. When the Areas of Expertise have stated the compliance to each requirement, they send the RFI reply back to the coordinator. The coordinator reviews the answers and sends the replies on to the Bid Support Specialist, who also checks the answers.

If the RFI originates from a major operator, a meeting is held with the Global Product Management, the coordinator, and experts from the Areas of Expertise, in order to discuss the answers which are to be submitted back to the operator. The RFI reply is then sent back to the operator by the Key Account Manager.

The RFIs play an important role in the operator's strategic planning. The RFIs also provide SEMC with vital business intelligence information as the features prioritized by the operators may be used as a guideline when developing future phones. The operators thus have a great deal of influence on the final requirements for a product and a good relationship with the operators, based on timely and correct replies to the RFIs, is therefore of utmost concern.

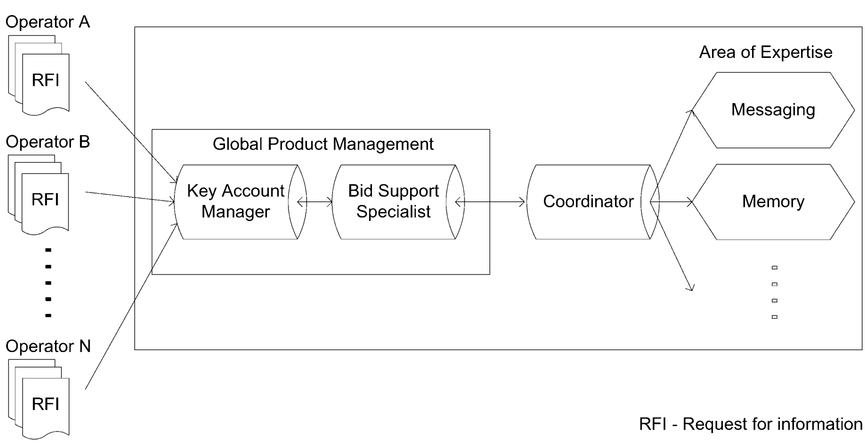


Fig. 2 The request for information process at SEMC

The efficiency of the RFI process, in which requirement are analyzed and checked against product features, is however severely impeded. The Areas of Expertise are concerned with their primary assignments in development and testing and have trouble finding the time required to analyze the RFIs. Furthermore, they get particularly frustrated as they have to state the compliance to the same or very similar requirements over and over again. Large parts of the new versions of RFIs arriving from the same operator are typically the same as previous versions. Furthermore, it is often the case that the same and very similar requirements appear in the RFIs from different operators.

Unfortunately, the requirement identifiers for the same requirements in two consecutive version of an RFI from the same operator may differ and can therefore not be used for simple consolidation. In general, the revision history of the RFIs cannot be relied upon as the operators' requirements processes are not suited or adapted to the activities that SEMC must conduct. Unfortunately, there are not enough financial incentives for the operators to make the required changes to their processes.

Consequently, there is much unnecessary redundant work required by the Areas of Expertise, and SEMC considers it an important goal to resolve the process bottlenecks. The following sections present the experiment that was designed to capture and investigate a potential solution to support the described work situation.

4. Experimental Conception

SEMC and many other market-driven companies (Karlsson et al., 2002; Mich et al., 2004) specify their requirements in natural languages. Previous research has shown that linguistic engineering may give valuable support in other large-scale requirements management activities, such as duplicate identification and linkage between customer and business requirements (Natt och Dag et al., 2002, 2004). That research, together with the problem description in the preceding section, gave incentives for conducting a controlled experiment for investigating if there is any significant gain from using linguistic engineering techniques to support requirements consolidation.

A laboratory experiment was design to capture the essence of the consolidation problem at SEMC and for evaluating a potential solution that could support the activity. A tool was developed for the purpose of giving support in the consolidation process. The tool takes two requirements sets as input and presents suggestions on similar requirements by calculation the fraction of words the requirements have in common (more information about the tool is found in the next section).

Figure 3 depicts the conceptual solution of the consolidation activity. To the left in the figure, two requirement sets, A and B, are shown. Suppose that the two sets represent two consecutive submissions of RFIs from the same operator. The earlier RFI, lets assume it is A, would already have been analyzed and the result from the analysis should be available in the central requirements database. The coordinator then uses the support tool to find requirements in B that are already analyzed in A and to mark them by assigning a link between them. The output of the process is shown to the right in the figure. The subset A' comprises all requirements that were previously analyzed but are no longer requested by the operator. Recall that the overlap is neither provided by the operator nor easily found through requirements

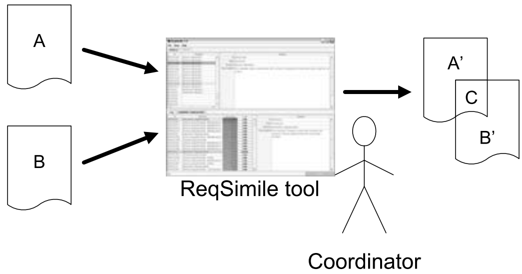


Fig. 3 The process of using the support tool for requirements consolidation

identifiers (see Section 3.1). The subset B' represents all new requirements that have not previously been analyzed. Finally, there is the subset C, which comprises all requirements in the new RFI that previously have been analyzed, i.e., these are the interlinked requirements. The coordinator would then send the requirements in set B' to the Areas of Expertise for analysis. The Areas of Expertise are thus relieved from the burden of re-analyzing the requirements in subset C.

5. Experimental Preparation

To minimize costs and spare resources at SEMC, the first level of experimentation was chosen, i.e., experimentation in a laboratory environment (Juristo and Moreno, 2001). Positive results from a first experiment may justify further experimentation in real projects. The rest of this section describes the preparation needed to conduct the experiment and the subjects acting in the experiment.

5.1. Subjects

The experiment was conducted in an academic environment and the experiment subjects were students taking a course in requirements engineering.[1] The course is given in the final year of a 3-year bachelor-level education program in software engineering. The students are thus soon to become professional practitioners in software engineering. The course gives 5 credit points corresponding to a quarter of a semester and the duration of the course is 8 weeks. The course includes the following parts:

– lectures based on the textbook by Lauesen (2002),
– 6 open-ended home exercises, each handed in by a group of students followed by classroom discussions session regarding the assessment and alternative solutions,
– a practical project in cooperation with local industry involving real RE problems with real stakeholders,
– and two hands-on lab session; one on prioritization using a commercially available RE tool and, this time, one on the requirements consolidation problem described in this paper.

[1] The course code is ETS671 and course information is available at http://serg.telecom.lth.se/education/.

The experiment operation and data collection was planned to be a part of the lab session, which was scheduled in the 4th week of the course. Before the data collection, the students had been taught requirements engineering terminology and have gained experience from the practical project. This is believed to be enough knowledge and experience in order to understand and carry out the experiment tasks.

The student population were between 21 and 30 years old with an average age of 23 years and there were 1 female and 22 male students. Further characteristics of the population, based on a pretest, are given in Section 6.3.

5.2. Tools

Two adapted versions of the ReqSimile tool (Natt och Dag et al., 2005), developed by the first author, were used in the experiment. The tool uses linguistic engineering to calculate similarity between requirements by using lexical similarity as a way of approximating semantic similarity. On a lexical level, we consider a requirement as a sequence of words. We process the sequence to drop words that have a purely grammatical role and to remove affixes and other lexical components not needed for comparison purposes. We then use the Cosine similarity measure as it does not depend on the relative size of the input. We also consider the weight of the words in the requirements (the number of times each word occurs). The result from calculating the Cosine measure is a number between 0 and 1 that indicate the similarity between the requirements based on the words they have in common. For details on the calculation steps, see (Natt och Dag et al., 2004, 2005; Natt och Dag and Gervasi, 2005).

The user interfaces of the two versions of the tool, ReqSimileA and ReqSimileM, are shown in Figs. 4 and 5 respectively. The left side in the top pane of the window presents a brief list of requirements. Selecting a requirement (1) makes the requirement's details show on the right (2) and a list of similar requirements in the other set appear in the bottom pane, sorted on the similarity value (3).

By clicking any of the Link buttons (4), a link is assigned between the selected requirement (2) and the requirement with the associated link button. The link is stored in an external database. A link may be removed by clicking the Unlink button, and the program will update the database accordingly. Linked requirements are highlighted according to a color scheme. All other requirements that are unlinked are uncolored (8).

ReqSimileM was modified so that it did not provide an automatically calculated list of similar requirements. Instead, it provided the possibility to enter search terms (9), which are used to retrieve similar requirements. The person using ReqSimileM must consequently come up with relevant search terms by him/herself. The keyword search support was provided to mimic the search facilities available in requirements management tools,[2] while, for experimentation purposes, keeping the differences between the versions as small as possible. Actually, ReqSimileM provides better support than most available requirements management tools as it sorts the resulting list of requirements based on the occurrence of keywords. Furthermore, the

[2] There is actually one tool, Focal Point (http://www.telelogic.com), which does provide functionality to find similar requirements based on the research presented in (Natt och Dag et al., 2002).

keywords are first preprocessed to remove inflections, etc., in the same manner as is done with the requirements in ReqSimileA.

Both versions of the tool were preset to use the requirements prepared as described in the next section. The original tool allows a user to select a requirement from either of the two sets in order to be presented with a list of similar requirements from the other set. This is done by clicking one of the tabs in the top half of the window named ‘PUSS A’ and ‘PUSS B,’ which are the names of the

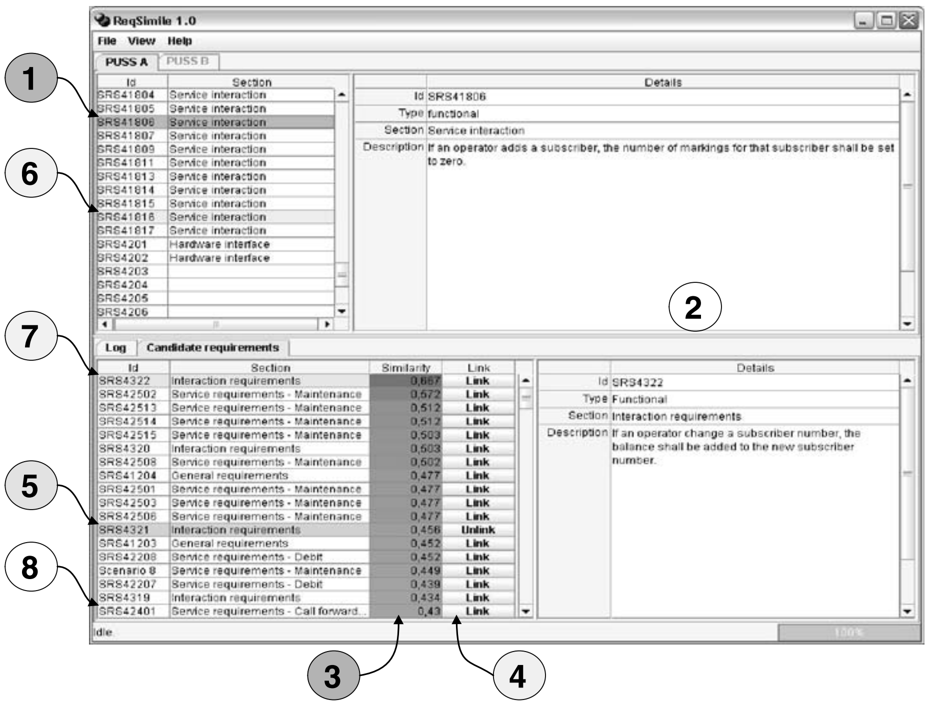


Fig. 4 ReqSimileA—providing automatic support

① **SELECTED REQUIREMENT** Currently selected requirement for which details are shown on the right and candidate requirements are shown on the bottom.
② On the right hand side the details are shown for a requirement that has been selected on the left.
③ The requirements are sorted on similarity to the above *selected requirement*. The fields are colored with a shade between red and white, representing the similarity. Red, or 1, represents an exact match.
④ This column has buttons for linking. **Link** is pressed to link and **Unlink** to unlink.
⑤ **LINKED REQUIREMENT 1** This requirement has been linked to the currently selected requirement above.
⑥ **LINKED REQUIREMENT 2a** This requirement has been linked. If you select it, the links may be reviewed using the bottom pane.
⑦ **LINKED REQUIREMENT 2b** This requirement has been linked, but not to the currently selected!
⑧ **UNLINKED REQUIREMENTS** White background means unlinked requirements. To link a requirement to the currently selected requirement, press the link button.
⑨ **SEARCH TERMS** Search terms are entered to fetch requirements that match the search terms. Multiple terms are separated by space. If no search terms are entered, all requirements are shown.

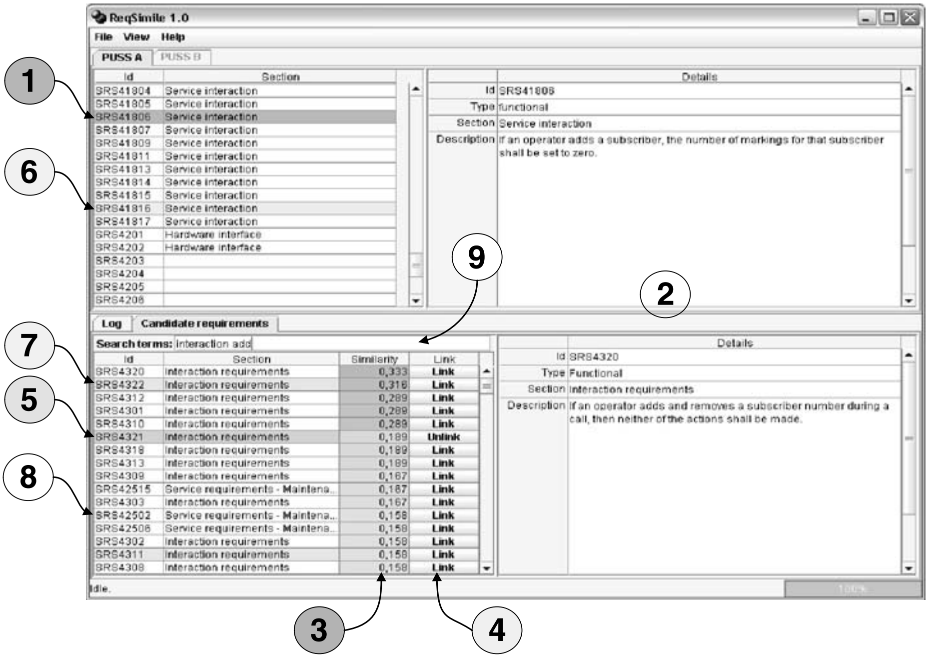


Fig. 5 ReqSimileM—providing manual key word search support

requirements specifications. However, to assure that the two experiment groups conducted the consolidation task in the same manner, both version of the tool were locked to enable selection from only one of the sets.

5.3. Requirement Specifications

For the consolidation experiment we used requirements specifications produced by students from the Master's degree program. The requirements specification had been produced as part of a course "Software Development for Large Systems,"[3] abbreviated PUSS (from the Swedish name of the course). The course comprises a full development project including requirements specification, test specification, high-level design, implementation, test, informal and formal reviews, and acceptance test. At the end of the course, the students deliver a first release of the controller software for an Ericsson switch board.

The requirements specifications, PUSS A and PUSS B, were chosen with respect to the experiment participants' knowledge in the domain. They had themselves taken the course one year earlier and were thus familiar with the requirements on such a system. Giving the course for 12 years, we have seen that although the students are developing the same system and are guided to write their requirements in one particular way, they naturally write their requirements differently. Furthermore, in order to make the projects slightly different, partly in order to avoid the

[3] The course code is ETS032 and course information is available at http://serg.telecom.lth.se/education/.

possibility of cheating, they are requested to develop different sets of services, e.g., redial, call forwarding, and take call. This results in different sets of requirements on two levels. First, the requirements required for each service and, secondly, the requirements resulting from resolving interaction issues between the services, e.g., in what way it is possible to take a call that has been forwarded to another subscriber.

Two specifications were randomly selected from the course given in the year 2002 and 2003. Note that the specifications were taken from the Master's degree program whereas the subjects were following the Bachelor's degree program. The programs are given in different cities, about 55 km apart. The requirements had been specified in the use case style or the feature style (Lauesen, 2002) and all requirements were written using natural language. In order not to burden the students (who produced the specifications) with domain terminology challenges in both Swedish and English, they had been recommended to write their specifications in Swedish (they have challenges enough to handle in the project). The two specifications, respectively, comprised 139 and 160 unique requirements and scenarios (see also Section 6.3 for more about the requirements used in the experiment). Tables 1 and 2 show example requirements.

As our tool expects the requirements to be written in English (the underlying linguistic engineering techniques are adapted for the English language) the requirements specifications were independently translated to English by the first two authors. No discussion was allowed between the translators during the trans-

Table 1 Example requirements from specification comprising 139 requirements

Key	Id	Type	Section	Description
3	Scenario 13	functional	Service: Regular call	Regular call-busy Actors: A:Calling subsriber, B:Called subscriber, S:System Prerequisites: Both A and B are connected to the system and are not unhooked. Step 13.1. A unhooks. Step 13.2. S starts giving dial tone to A Step 13.3. A dials the first digit in B's subscriber number Step 13.4. S stops giving dial tone to A. Step 13.5. A dials the remaining three digits in B's subscriber number Step 13.8. S starts giving busy tone to A Step 13.9. A hangs up Step 13.10. S stops giving busy tone to A
80	SRS41606	functional	Service: Call forwarding	Activation of call forwarding to a subscriber that has activated call forwarding shall be ignored by the system. This is regarded as an erroneous activation, and an error tone is given to the subscriber. (Motivation: Together with SR41607, avoids call forwarding in closed loops)
111	SRS41804	functional	Service interaction	The service call forwarding shall be deactivated if an operator removes either the subscriber from which calls are forwarded or the subscriber to which calls are forwarded.

Table 2 Example requirements from specification comprising 160 requirements

Key	Id	Type	Section	Description
41	Scenario 13	Functional	Service requirements-Normal call	"Normal call-busy" Actors: A Calling subscriber. B: Subscriber which are called to. S: System. Condition: Both A and B are connected. A's receiver is on hook. B's receiver is off hook. Step 13.1–13.5. [The same as 11.1–11.5] Step 13.6. S gives busy tone to A. Step 13.7. A on hook. Step 13.8. S stops busy tone to A.
91	SRS42403	Functional	Service requirements-call forwarding	If call forwarding is done to a subscriber number which in its turn is forwarded, an error tone shall be used.
125	SRS4306	Functional	Interaction requirements	If a subscriber number is forwarded to another subscriber number and the later is removed, then the call forwarding shall be cancelled.
130	SRS4311	Functional	Interaction requirements	If a forwarded subscriber is removed, then the call forwarding shall be cancelled.

lation in order to reduce internal validity threats. In particular, we wanted to preserve the possibilities of using different wordings and injecting spelling errors. However, no errors or differences between the requirements sets were deliberately injected.

The tool expects the requirements to reside in a database so the requirements' content was transferred to fields in a database table. The resulting database table structure and example requirements can be found in Tables 1 and 2. The two tables show requirements from the requirements specifications that are semantically similar. The examples also show the varying linguistic quality and the presence of differences between similar requirements. The two scenarios refer to the same underlying functionality. Note the error in the scenario in Table 1, where it says that no subscriber initially is unhooked (compare to the other scenario). Also note the left-out steps in the scenario in Table 2. Requirements SRS41606 in Table 1 and SRS42403 in Table 2 are similar but are written differently. Requirement SRS41804 in Table 1 refer to the same functionality as requirements SRS4306 and SRS4311 in Table 2 (together, i.e., a split requirement). Notice the alternate spelling of *canceled*.

In this experiment, the content from the *section* and the *description* fields were used to calculate the similarity between requirements. For more information about the underlying calculation procedure see (Natt och Dag et al., 2004).

5.4. Correct Consolidation

To enable measurement of the subjects' accuracy of linking requirements that are semantically similar, the first author created a key in which the correct links had been assigned. The author has been teaching in the course for several years, taking different roles in the course, such as requirements expert, test expert, and expert reviewer, as well as holding exercises and laboratory sessions. It is therefore justifiable to regard the key as one provided by an expert in the domain. The key was created prior to any analysis of the subjects' assigned links in order to reduce any related validity threats.

A way of measuring the tool's ability of calculating similarities between requirements is to see at which positions the correctly similar requirements end up in the ranked list produced by the tool. The requirements that the expert considers to be similar should preferably appear at the top of the list. Of the 30 requirements that the subjects were asked to consolidate (see Section 6.3), 20 requirements were considered similar by our expert analysis.

In Fig. 6, a histogram is found showing the distribution of the positions at which the correctly similar requirements end up in the ranked lists produced by the tool (see Section 5.2). As shown in the figure, almost all (17 out of the 20) ended up at position 8 or better in the ranked list and should therefore quickly be spotted. For presentation purposes, one data point is absent in the figure. One similar requirement was found at position 64 in the produced top list. We found two reasons for this. Firstly, vital information was available in another requirement (i.e., the requirement was split). Secondly, the differing lengths of the requirements seems to affect the similarity calculation adversely. The similarity measure itself is less sensitive to the lengths of the requirements, but shorter requirements contain less information to match.

6. Experimental Planning

In this section, the variables, the hypotheses, and the design of the experiment are described. In addition, the threats to validity are discussed.

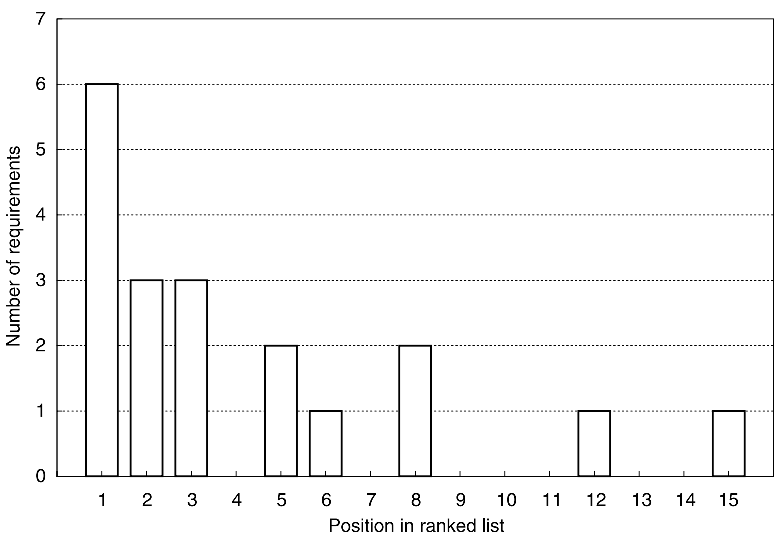


Fig. 6 Histogram of the positions of similar requirements in the automatically produced ranked list of similar requirements

6.1. Variables

The variables in the experiment are described below, grouped into independent, controlled, and dependent.

- The independent variable is the methods used in the experiment. The two methods compared are manual and assisted.
- The controlled variable is the experience of the participants. A questionnaire was used prior to the experiment to analyze the individual experience of the subjects.
- The dependent variables are:

  T time used for the consolidation
  N the number of analyzed requirements
  N_{cl} number of correct links
  N_{il} number of incorrect links
  N_{cu} number of correctly not linked
  N_{iu} number of missed links (incorrectly not linked)

  These dependent variables are used to analyze the hypotheses. The number of analyzed requirements are used in case the subjects are not able to analyze all requirements, which will affect N_{iu} and N_{cu}.

6.2. Hypotheses

The hypotheses of the experiment were defined to investigate whether the requirements engineers can be aided by using the assisted method. Presented below are six null hypotheses. The *assisted method* refers to the method used by subject group that used the ReqSimileA tool. The *manual method* refers to the method used by the subject group that used the ReqSimileM tool (see Section 5.2 for a description of the tools).

H_0^1 The assisted method results in the same number of *requirements analyzed per minute*, N/T, as does the manual method.
H_0^2 The assisted method results in the same share of *correctly linked requirements*, $N_{cl}/(N_{cl}+N_{iu})$, as does the manual method.
H_0^3 The assisted method results in the same share of *missed requirements links*, $N_{iu}/(N_{cl}+N_{iu})$, as does the manual method.
H_0^4 The assisted method results in the same share of *incorrectly linked requirements*, N_{il}/N, as does the manual method.
H_0^5 The assisted method is as *precise*, $N_{cl}/(N_{cl}+N_{il})$, as the manual method.
H_0^6 The assisted method is as *accurate*, $(N_{cl}+N_{cu})/(N_{cl}+N_{il}+N_{cu}+N_{iu})$, as the manual method.

In this experiment we are interested only to find out if the assisted method outperforms the manual. It is of no practical importance if the assisted method is less effective than the manual method. Therefore, we want to reject the null hypotheses only when the difference is in the expected direction. Therefore, we define the following one-sided alternative hypotheses:

H_1^1 The assisted method results in a greater number of *requirements analyzed per minute*, N/T, than does the manual method.
H_1^2 The assisted method results in a larger share of *correctly liked requirements*, $N_{cl}/(N_{cl} + N_{iu})$, than does the manual method.
H_1^3 The assisted method results in a smaller share of *missed requirements links*, $N_{iu}/(N_{cl} + N_{iu})$, than does the manual method.
H_1^4 The assisted method results in a smaller share of *incorrectly linked requirements*, N_{il}/N, than does the manual method.
H_1^5 The assisted method is more *precise*, $N_{cl}/(N_{cl} + N_{il})$, than the manual method.
H_1^6 The assisted method is more *accurate*, $(N_{cl} + N_{cu})/(N_{cl} + N_{il} + N_{cu} + N_{iu})$, than the manual method.

6.3. Experimental Design

A pilot experiment was performed to evaluate the instrumentation and design. Four colleagues participated and used the methods evaluated in the experiment. After the pilot experiment, it was concluded that the number of requirements should be reduced in order to enable the subjects to consolidate an expected number of requirements within a lab session. Therefore, 30 requirements were randomly selected from the set of 139 requirements (see Section 5.3 about the requirements). The same set of 30 requirements were given to each subject during the experiment.

Before the main experiment, a questionnaire (pre-test) comprising 6 questions was handed out to the students to explore their experience from industrial software development and course participation, and skills in English reading and writing. The questionnaire showed that they had equal industrial experience (none) and equal experience from earlier courses. The differences in reading and writing skills were small, but were used to randomly assign the participants to two experiment groups. This resulted in 11 students in the subject group that were given the ReqSimileM tool (groupM) and 12 students in the subject group that were given the ReqSimileA tool (groupA). One student in groupM and two students in groupA were removed from the analysis, since they did not perform all parts of the experiment. Thus, each group had 10 subjects. The experiment was carried out during two hours, which included a short introduction to the task, the experiment, and a post-test.

The experiment data has been analyzed with descriptive analysis and statistical tests (Montgomery, 2001). The data was checked for normal distribution (normal probability plots), and the conclusion was that parametric hypothesis tests could be used. The two-sample one-tailed t-test was used to investigate the hypotheses (Montgomery, 2001). The significance value for rejecting the null hypotheses was set to 0.05 for all tests.

6.4. Threats to Validity

The result of an experiment should be interpreted in the light of the threats to the validity, in order to help future replications, and generalization of the result. In this section, the threats are analyzed related to four groups of threats: Conclusion validity, internal validity, construct validity, and external validity (Wohlin et al., 2000).

Conclusion validity concerns the relation between the treatments and the outcome of the experiment. The threats related to the statistical tests used in the experiment are considered being under control since normal probability plots have been used to check that parametric tests can be used. These have greater power that non-parametric, which is advantageous since only 23 subjects participated in the experiment. Threats with respect to the subjects are also limited since the subject groups are rather homogeneous, the subjects have attended the same education program during 2.5 years.

Internal validity of the experiment concerns the question whether the effect is caused by the independent variables or by other factors. The experiment has limited threats due to history, maturation, mortality, etc., since it is applied during a 2 hour period. The social threats are limited since the subjects had nothing to gain from the actual outcome of the experiment. The grading of the courses was only based on their participation in the experiment, not on their performance. The threat of selection is also under control, as the experiment is a mandatory part of a course. However, three subjects were considered as outliers since they did not make their best effort during the experiment session, see Section 8.

A potentially more problematic threat is that the subjects had to analyze and link requirements written in English when they had themselves used only Swedish to specify their own requirements in the domain. This threat were minimized through the pre-test where we asked about their ability to read and write common and technical English. Although the questions were answered through self-assessment, the students' answers that they generally can read and write both common English and technical English quite easily or fluently, is credible. Swedish students generally speak and write English well and a major part of their course material is in English. Furthermore, our own experience from translating the specifications suggests that it is easier to understand the Swedish concept based on the English written one, than to translate to the correct English term from Swedish.

Construct validity concerns the ability to generalize from the experiment result to the concept behind the experiment. There are two potential threats, experimenter expectations and interaction of testing and treatment. The main experimenter developed the tool, which is the basis of his research. Hence, he expected the tool to be better than the manual method. To reduce this threat, two other people were included as experimenters (planning, operation and analysis), who are not directly involved in the research of the main experimenter. Furthermore, since two measures in the experiment are the number of correct and faulty links, the subjects may have been more aware of their errors. Also, when the subjects know that the time is measured, it is possible that they get more aware of the time they spend, and thus the time consumption is affected. Subject's awareness cannot be controlled, but the analysis does not indicate any major impact.

The experiment would need another set of requirements in order to enable us to discover whether the results are the same or if the set of requirements have affected the results.

External validity concerns the ability to generalize the results to industry practice. The largest threat is that only a few requirements have been used in the experiment. With larger sets, the automatic similarity calculation may not result in high ranking of actually similar requirements. Earlier research investigates this with larger sets (Natt och Dag et al., 2002, 2004) and shows that the techniques are good enough to give valuable support. Nevertheless, further experimentation is needed.

Another large threat is that students are used as subjects. However, the students are in their third year of software engineering studies and thus close to start working in industry. The participants are familiar with the domain (software development), which is industry-like, as they participate in a requirements engineering course. Furthermore, as most experimental conditions, the time is an important factor. In order to reduce the fatigue effect, the number of prioritized requirements are fewer than in most real cases. Thus, it is difficult to judge whether extending the number of requirements would lead to the same result.

In summary, the threats are not considered large in this experiment. The main threats are that fewer requirements were used than in a real case and that students were used as subjects. Hence, future replications and case studies need to be performed in order to reduce these threats.

7. Experimental Operation

The experiment was run in one two-hours lab session in the fall of 2004. The first 15 minutes of the 90 minutes lab session was dedicated to a presentation by the first author of the general problem, the industrial applicability, and the goal of the lab session. All students were given the same presentation, which included also an overview of the generic aspects of the two versions of the tool. The general differences between the tools were communicated without favoring one over the other. To avoid biasing, no hypotheses were revealed and it was made very clear that we did not know whether one of the approaches was in any way superior over the other.

After the presentation, the students were separated into two groups depending on which version of the tool they were going to use during the experiment. The two groups were directed to separate rooms and instructed to prepare for and begin the task as written in the material. Each group was accompanied by a supervisor. No student was allowed to discuss his or her work with another student during the experiment. However, they were allowed to ask questions to the supervisor if they experienced any problems. Some students asked questions, which were mainly about how two requirements could be regarded as similar or not. Direct answers to these questions were not given. Instead we referred to the definitions given in the material and rephrased or clarified these when needed.

The material used in the experiment comprised:

1. The ReqSimile package. This had been put into a zip file, which was downloadable from the course home page. Depending on the group, the students were asked to download one of the packages and run the application. The package included:
   a. The ReqSimile application, either with automated support or for manually specifying search terms.
   b. MS Access database including
      - 30 randomly selected requirements from PUSS A. These could be browsed and were to be linked to requirements in PUSS B.

- All 160 requirements from PUSS B. These could be browsed through the suggestion list which was, depending on the tool version, presented upon selection of a requirement from PUSS A or upon entering search terms in the search text box.
- Preprocessed requirements. Separate tables to speed up the similarity calculation. For the full set of requirements the preprocessing stage took only 3 seconds, but it was completed prior to the experiment to minimize the complexity of the task.
- Empty link table in which the links set by the students would be stored.

2. A document including:

   a. Industrial scenario describing the actual challenge
   b. A general task description
   c. Detailed tasks with space for noting down start and end times.
   d. A short FAQ with general questions and answers about the requirements
   e. A screen shot of the tool user interface with descriptions of the different parts.

   The instructions to the students were:

– Walk through as many of the requirements as possible of the 30 shown in the tool. For each requirements, investigate if there are any requirements in the other set that can be considered identical or very similar (only differing in some detail) with respect to intended functionality.
– Assign a link between requirements that are considered identical or very similar.
– Log the times for start and finish.
– When finished, notify the supervisor.

The subjects were initially given 60 minutes for the consolidation task, but due to operating system difficulties (promptly resolved by the subjects) the subjects had about 45 minutes to consolidate the requirements.

15 minutes before the end of the lab session, the students that had not analyzed all 30 requirements were asked to stop working. All students were then given a post-test document. In the post-test document they were asked to note down the finishing time and the number of requirements they had analyzed.

8. Analysis

This section presents the data collected during the experiment and the results from the statistical analyses used to investigate if any of the null hypotheses could be rejected. Each of the different hypotheses will be addressed separately and conjectures will be made iteratively based on the accumulated results from the hypothesis tests. All the measurements of the dependent variables are found in Table 3. It should be noted that one analyzed requirement can be linked to several others. Therefore, it is not possible to see any functional relationship between the number of analyzed requirements and the other measures.

Table 3 The collected experimental data

Subject	T (min)	N	N_{cl}	N_{cu}	N_{il}	N_{iu}
A1	47	27	11	7	18	6
A2	44	17	7	5	12	3
A3	46	25	9	8	12	7
A4	46	17	8	5	12	2
A5	40	30	8	11	1	12
A6	37	17	6	4	8	4
A7	29	3	1	0	6	0
A8	30	3	1	0	3	0
A9	45	13	5	3	8	3
A10	47	22	12	5	9	1
A11	45	21	10	5	11	2
A12	35	30	14	7	11	6
M2	43	19	7	6	6	5
M3	42	19	6	5	6	6
M4	43	15	2	3	10	7
M5	45	15	6	2	10	3
M6	45	20	4	6	6	8
M7	45	13	4	3	17	4
M8	45	15	4	5	2	5
M9	45	16	5	5	3	4
M10	45	22	9	6	11	4
M11	40	15	3	6	1	6
M12	60	4	0	3	2	1

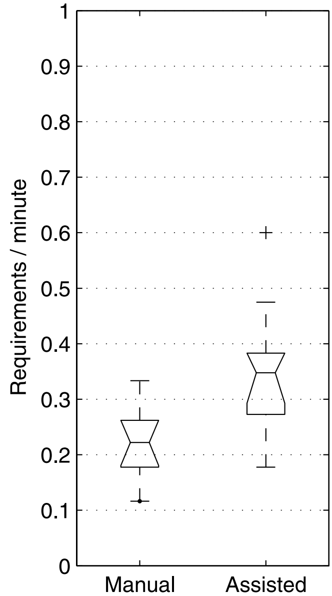
8.1. Performance

The background to the experiment and the industrial case shows that it is a time-consuming task to consolidate requirements. Therefore, one of the first interesting collected measurements is the number of requirements the subjects were able to analyze during the time of the experiment.

The subjects did not use the same amount of time so we calculated the number of requirements they were able to analyze per minute. This would be an appropriate measure of their performance. Figure 7 shows the results and it is clear that there is a significant difference between the manual method and the assisted method. GroupA analyzed on average 0.35 requirements per minute (2.8 minutes per requirement), while groupB analyzed in average 0.22 requirements per minute (4.5 minutes per requirement). In this case, the means and the medians coincide. The notches of the box plot gives a graphical representation of the significance of the difference between two means. In all the presented box plots the means of the data sets are significantly different at approximately the 0.05 level if the box plot's notches do not overlap.

In this particular case it is possible to reject the null hypothesis H_0^1 at a level of significance below 0.004. Thus, using the assisted method provides a significantly greater performance than using the manual method. This is a welcomed result, as we are interested in saving time in industry. However, further analysis is required before we may draw any final conclusions about the approach.

Fig. 7 Analyzed requirements per minute



8.2. Quality

Naturally, nothing can be said whether the assisted method is actually better than the manual if the quality of the subjects' results is not evaluated. Three hypotheses have been formulated to decide if the assisted method enables the subjects to assign more correct, fewer missed, and fewer incorrect links (Section 6.2).

To test the null hypothesis H_0^2, we compared each subject's assigned links with the correct answers produced from our expert analysis. The result is shown in Fig. 8. Again, it shows a difference between the manual and assisted method. GroupA correctly assigned on average 68% of the links that the expert assigned (the median is just 0.6% less), while groupB correctly assigned on average 48% of the links (the median is, as can be seen in the figure, 50%).

The t-test shows that we can reject the null hypothesis at a significance level below 0.005. We may thus conclude that the assisted method enables the subjects to correctly link a significantly larger number of requirements than if they had used the manual method. Together with the performance statistics, this suggests that the assisted method is superior to the manual. However, other quality aspects exist.

For hypothesis H_0^3, we looked at each subject's analyzed requirements and counted how many links that should have been assigned, but were not. Since the subjects did not analyze the same number of requirements, it is important to only consider the requirements they had been able to analyze. The subjects had stated in the post-test how many requirements they had analyzed and they had worked through the requirements one after the other. This enabled us to correctly count the number of missed links compared to our expert analysis. Figure 9 shows that there is a significant difference. GroupA missed on average 32% of the links that the experts assigned (median is about 0.5% higher), while groupB missed on average 52% of the links (median is 50%).

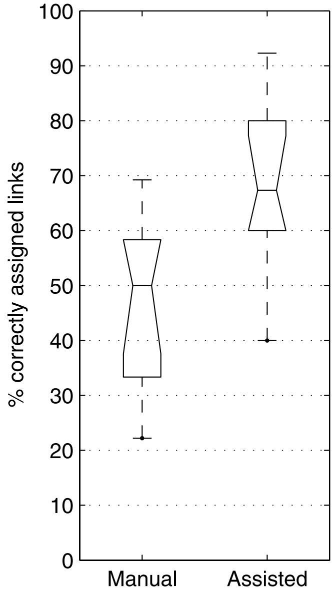


Fig. 8 Correctly linked requirements

The t-test also reveals that we may reject the null hypothesis on a significance level below 0.005. This enable us to make the conjecture that the assisted method helps the subjects to miss significantly fewer requirements links. This makes the case only stronger; that the assisted method is to be preferred over the manual.

A conclusive quality measure would be the number of links that have been incorrectly assigned between two requirements. In Fig. 10, the number of incorrectly assigned requirement for the two subjects groups are shown. It can be noted from

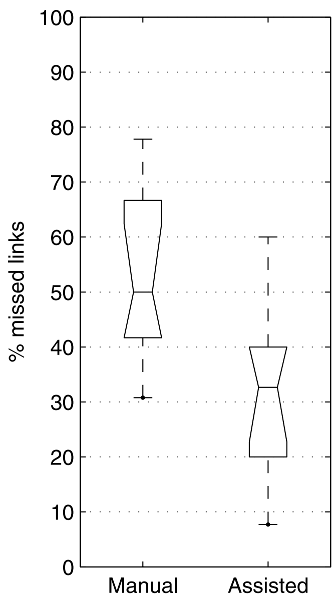
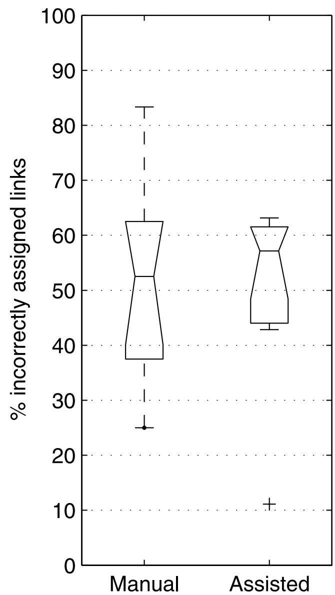


Fig. 9 Missed requirements links

Fig. 10 Incorrectly linked requirements



the figure that the means are not significantly different (In average 51% for groupA and 53% for groupB, with medians 52% and 57% respectively). A t-test also acknowledge that we are not able to reject the null hypothesis H_0^4. Although the figure indicates that the subjects using the assisted method actually assign more incorrect links, there is no statistically significant difference.

How does the last statistic affect our case? In the consolidation process it would be very unfavorable to say that two requirements are similar when they are not, because those requirements might not be further analyzed and the mistake may be discovered too late. The error is human and must be addressed in other ways. One way is to raise the threshold for linking two requirements, meaning that if the human analysts are only the slightest hesitant of the similarity between two requirements, they should let it be. The requirement will the be sent to the Area of Expertise who makes a proper analysis.

8.3. Precision and Accuracy

There is another way to assess the quality of the subjects performance by using the concepts of precision and accuracy. Precision is a measure that takes into account both the correctly and incorrectly linked requirements. Using an analogy, it is a measure of how much weed we have in the harvest. Accuracy, then, is a measure of how well the result conforms to the truth. The truth in this case would be to assign all links correctly, not miss any, and not link anything that should not have been linked.

Unfortunately, both precision and accuracy incorporates the number of incorrectly linked requirements, which has a great impact on the measures. The number of incorrectly linked requirements are comparatively large in relation to the other measures (see Table 3). Therefore, it is not likely that we will find any significant differences between the two groups. The statistics of the incorrectly linked requirements propagates to the statistics of these measures. And quite so, our

Table 4 Summary of the results of the hypotheses

Hypothesis	p-value
H_0^1, Speed	0.0034 (significant)
H_0^2, Correct links	0.0047 (significant)
H_0^3, Missed links	0.0047 (significant)
H_0^4, Incorrect links	0.39
H_0^5, Precision	0.39
H_0^6, Accuracy	0.15

analysis of precision and accuracy of the two subject group shows no statistical differences. Consequently, we are unable to reject both hypothesis H_0^5 and H_0^6.

9. Discussion

The analysis of the experiment data allows for the rejection of three out of six null hypotheses (Table 4). Although it can not be statistically determined if there are any differences in precision and accuracy, the other results point in the same direction: that the assisted method is superior to the manual.

Previous investigations by Natt och Dag et al. (2002, 2004) suggests that linguistic engineering techniques may give support in certain requirements management activities. These activities (duplicate identification and linkage of customer wishes and business requirements) rely on an underlying activity which they share with the requirements consolidation process: that of finding and matching similar requirements. Previous research suggests that the proposed linguistic engineering techniques may support this activity and that time could be saved in requirements management. The results from this experiment gives further evidence to this claim.

The results from this experiment are also supported by initial evaluations made at SEMC. In parallel with the operation and analysis of this experiment, two other students have worked on the ReqSimile tool, adapting it to the specific requirements management process at SEMC and to the requirements analysts' needs. The initial evaluation with experts is very promising, indicating time-savings of up to 30−40% in the activities of consolidating and analyzing RFIs. The experts have also provided their subjective feedback on the tool, which is very positive. Further studies are motivated, but that even initial results from a real industrial setting point in the same direction as the results from the presented experiment, makes the case somewhat stronger.

As always, when conducting experiments to increase the body of knowledge, the experiment should be replicated in different contexts. This experiment has been conducted in a laboratory setting and it would now be interesting to use experts as subjects in a real project.

Improvements to the tool and the underlying techniques are of course possible. Different models for calculating similarity are possible. This is elaborated further in related work (Natt och Dag et al., 2002, 2004, 2005; Natt och Dag and Gervasi, 2005).

10. Conclusions

In this paper we have presented an experiment with a tool that aims at giving support in a specific large-scale requirements management activities, namely requirements consolidation. A background and an industrial case have been presented to explain the origin of the requirements consolidation activity and to motivate the experiment. Experiences from an initial evaluation of the tool in a real industrial environment, indicating time-savings of up to 40%, have also been reported.

Two treatments were compared. One *assisted*, in which the subjects were presented with an automatically calculated ranked list of similar requirements, and one *manual* in which the subject had to come up with their own search terms to find similar requirements. The task for the subjects was to assign links between requirements that could be considered similar.

The main results from the experiment is that the subjects using the assisted method are able to assign more correct links and also do this faster. Furthermore, they miss fewer links than the subjects using the manual method. The important results from the experiment are:

Speed. The subjects using the assisted method are able to analyze significantly more requirements than the subjects using the manual method.
Correctness. The subjects using the assisted method assign significantly more correct links than the subjects using the manual method.
Missed links. The subjects using the assisted method miss significantly fewer links than the subjects using the manual method.

The presented experiment is an experiment at the first level of experimentation, i.e., conducted in a laboratory environment (Juristo and Moreno, 2001). The results from the experiment are promising for proceeding with both replications and with new experiments in a real industrial environment.

Acknowledgments The authors would like to thank Daniel Karlström, Per Runeson, Martin Höst, and Lena Karlsson for participating in the pilot experiment. The authors would also like to thank the students for participating in this investigation and Lena Karlsson for acting as a stand-in experimenter during the experiment session with the students. Finally, we would like to direct a special thanks to our anonymous reviewers for valuable comments.

References

Brinkkemper S (2004) Requirements engineering research the industry is (and is not) wating for. In: Regnell B, Kamsties E, Gervasi V (eds), Proceedings of the 10th Anniversary International Workshop on Requirements Engineering: Foundations for Software Quality, Essen, Germany: Essener Informatik Beiträge, pp 267–284

Clements P, Northrop L, (2002) Software product lines: practices and patterns. Boston, MA: Addison-Wesley

Hofmann HF, Lehner, F (2001) Requirements engineering as a success factor in software projects. IEEE Software 18(4):58–66

Juristo N, Moreno AM, (2001) Basics of software engineering experimentation. Boston, MA: Kluwer Academic Publishers

Karlsson L, Dahlstedt AG, Natt och Dag J, Regnell B, Persson A (2002) Challenges in market-driven requirements engineering—an industrial interview study. In: Saliensi C, Regnell B,

Pohl K, (eds), Proceedings of the Eigth International Workshop on Requirements Engineering: Foundation for Software Quality, Essen, Germany: Essener Informatik Beiträge, pp 37–49
Kotler P (2002) Marketing management. Upper Saddle River, NJ: Prentice Hall
Krueger CW (2003) Towards a taxonomy for software product lines. In: Proceedings of the 5th International Workshop on Product Family Engineering, Siena, Italy
Lauesen S (2002) Software requirements: styles and techniques. London, UK: Addison-Wasley
Lubars M, Potts C, Richter C (1993) A Review of the state of the practice in requirements modeling. In: Proceedings of IEEE International Symposium on Requirements Engineering, Los Alamitos, CA: IEEE CS, pp 2–14
Mich L, Franch M, Novi Inverardi PL (2004) Market research for requirements analysis using linguistic tools. Requirements Engineering 9(1):40–56
Montgomery DC (2001) Design and analysis of experiments. USA: John Wiley & Sons
Natt och Dag J, Gervasi V (2005) Managing large repositories of natural language requirements. In: Aurum A, Wohlin C (eds) Engineering and Managing Software Requirements, Springer-Verlag
Natt och Dag J, Regnell B, Carlshamre P, Anderssonm M, Karlsson J (2002) A feasibility study of automated natural language requirements analysis in market-driven development. Requirements Engineering 7(1):20–33
Natt och Dag J, Gervasi V, Brinkkemper S, Regnell B (2004) Speeding up requirements management in a product software company: linking customer whishes to product requirements through linguistic engineering. In: Proceedings of the International Requirements Engineering Conference (RE2004), Kyoto, Japan: IEEE CS, pp 283–294
Natt och Dag J, Gervasi V, Brinkkemper S, Regnell B (2005) A linguistic-engineering approach to large-scale requirements management. IEEE Software 22(1):32–39
Novorita RJ, Grube G (1996) Benefits of structured requirements methods for market-based enterprises. In: Proceedings of the Sixth Annual International Symposium on Systems Engineering (INCOSE'96), Boston, MA
Potts C (1995) Invented requirements and imagined customers: requirements engineering for Off-The-Shelf Software. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering, Los Alamitos, CA: IEEE CS pp 128–130
Regnell B, Beremark P, Eklundh O (1998) A market-driven requirements engineering process—results from an industrial process improvement programme. Journal of Requirements Engineering 3(2):121–129
Sawyer P, Sommerville I, Kotonya G (1999) Improving market-driven RE Processes. In: Proceedings of International Conference on Product Focused Software Process Improvement (PROFES'99), Jun, Oulu, Finland, pp 222–236
Sommerville I (2001) Software Engineering. 6th edition. Harlow, UK: Pearson Education
Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in software engineering—an introduction. Norwell, MA: Kluwer
Yeh A (1992) Requirements engineering support technique (REQUEST)—a market driven requirements management process. In: Proceedings of the Second Symposium on Assessment of Quality Software Development Tools, Los Alamitos, CA: IEEE CS, pp 211–223

Dr. Johan Natt och Dag is the Requirements Engineering Manager at Obigo AB (a Teleca company). His research interests included requirements engineering, software product management, market-driven software development, software quality, and usability egnineering. He received his Ph.D. in software engineering from Lund University. Contact him at Obigo AB, Scheelevägen 17, SE-223 70 Lund, Sweden; johan.nattochdag@obigo.com.

Dr. Thomas Thelin is an associate professor of software engineering at Lund University. His research interests include empirical methods in software engineering; software quality, and verification and validation with emphasis on testing, inspections, and estimation methods. He received a Ph.D. in software engineering from Lund University. Contact him at Dept. of Communication Systems, Lund Univ., PO Box 118, SE-221 00 Lund, Sweden; thomas.thelin@telecom.lth.se.

Dr. Björn Regnell is an associate professor in software engineering at Lund University. His research interests include requirements engineering, empirical software engineering, software product management, process improvement, and market-driven software development. He received his Ph.D. in software engineering from Lund University. He is chair of the Swedish Requirements Engineering Research Network (SiREN). Contact him at the Dept. of Communication Systems, Lund Univ., PO Box 118, SE-221 00 Lund, Sweden; bjorn.regnell@telecom.lth.se.