Requirements Eng (2002) 7:20–33
Ownership and Copyright

Requirements
Engineering

A Feasibility Study of Automated Natural Language Requirements Analysis in Market-Driven Development

Johan Natt och Dag[a], Björn Regnell[a], Pär Carlshamre[b], Michael Andersson[c] and Joachim Karlsson[d]

[a]Department of Communication Systems, Lund University, Sweden; [b]Ericsson Radio Systems AB, Linköping, Sweden; [c]Telelogic Technologies AB, Malmö, Sweden; [d]Focal Point AB, Linköping, Sweden

In market-driven software development there is a strong need for support to handle congestion in the requirements engineering process, which may occur as the demand for short time-to-market is combined with a rapid arrival of new requirements from many different sources. Automated analysis of the continuous flow of incoming requirements provides an opportunity to increase the efficiency of the requirements engineering process. This paper presents empirical evaluations of the benefit of automated similarity analysis of textual requirements, where existing information retrieval techniques are used to statistically measure requirements similarity. The results show that automated analysis of similarity among textual requirements is a promising technique that may provide effective support in identifying relationships between requirements.

Keywords: Automated analysis; COTS; Duplicate identification; Natural language; NLP; Similarity

1. Introduction

1.1. Background

The market-driven development organisation faces many challenges that differ from those found in organisations developing bespoke software. Software is developed for a large market, rather than for a specific customer, new versions are developed in a succession of releases, and there is a high pressure on short time-to-market [1–3]. To meet market demands it is important to have an effective and efficient requirements engineering process. Special demands are set as requirements arrive continuously at a high rate from many different sources during the whole development process [4]. As there is no single specific customer to negotiate with, requirements must be invented within the developing organisation based on foreseen end-user needs [5]. These invented requirements may come from sources such as marketing, support, development, testing, usability evaluations and technology forecasting, and are often collected for storage in a database. The requirements engineering activities are then focused on analysing and prioritising the requirements in the database and on maintaining the database for the future.

In this study we have focused on a large software developing company, Telelogic AB, that develops a CASE tool for the worldwide telecommunications market. Their development process is described in Regnell et al. [4] and its main properties are:

1. Releases are pipelined to enable a new release every sixth month while each release takes 14 months to complete.
2. Elicitation is continuously active and a requirement may be issued at any time by an issuer that foresees a market need.
3. Each requirement is stored in a database as an entity described in natural language.
4. Each requirement has a life cycle progression through specific states.

The Telelogic development process has shown to have high resemblance to another market-driven development process independently developed and used at an Ericsson company [6].

Correspondence and offprint requests to: J. Natt och Dag, Dept. of Communication Systems, Ole Römers v. 3, 223 63 Lund, Sweden. Email: johan.nattochdag@telecom.lth.se

Requirements are continuously collected through a web form and are stored in a database for further analysis [4]. The requirements are described in natural language and are of varying quality and nature. Some requirements are brief ideas while others are detailed descriptions of new features with accompanying code. Many requirements are short-worded and poorly written.

During the development of a release the requirements engineer (or analyst) must handle the diverse and large set of requirements that is available in the database and resolve ambiguities, find relationships, eliminate duplicates, etc. As shown in a study of the Telelogic requirements process [7], these activities are causing a congestion that may be avoided by cutting down heavily on the number of elicited requirements or making early and strict prioritisation.

The trade-off between analysing only a subset of all the collected requirements and not collecting that many requirements to give time for proper analysis may be difficult to make. Extra information could be extracted if all requirements are collected (for example, duplicates may indicate that certain issues are more important than others). However, trying to handle all incoming requirements may increase the risk of relationships between requirements being overlooked or discovered too late, which may cause problems in prioritisation [8] and release planning [6].

Consequently, there is a wish to find requirements relations early, without spending too much time on in-depth analysis. These relationships should preferably be found even when specification quality is low and even if requirements are short, poorly worded or misspelled. One possible approach, investigated in this paper, is to assist the requirements engineer through automated analysis of the textual information in the requirements. This approach may help the requirements engineer to handle the large set of requirements by automatically finding and make suggestions on relationships between requirements.

Two different automatic text-processing approaches may be used to aid the requirements engineer in the situation described above: the statistical approach or the linguistic approach. In this paper we focus on the statistical approach, which originates from the work by H. P. Luhn [9]. There are several reasons that we choose to explore this approach:

1. The ideas have not, as we far as we know, been applied to analyse the type of requirements that is collected in the situation we describe (see further Section 1.2).
2. The statistical approach has been thoroughly tried and examined and has been found fairly successful for automatic text analysis [10].
3. The linguistic approach is still regarded as expensive to implement and not always more effective than well-executed statistical approaches [11].
4. Before proceeding with more advanced methods, the statistical approach may help reveal the nature of the requirements in a market-driven organisation.
5. A baseline produced from empirical investigation using real industry requirements is needed to compare against further improvements.

The results of the presented work show that, for a particular set of requirements, a simple similarity analyser that uses the statistical text-processing approach identifies a large fraction of the requirements duplicate pairs found by experts. The duplicates are important to find to avoid doing the same job twice, assigning the same requirement to different developers, or getting two solutions to the same problem. The portion of requirement pairs incorrectly identified as duplicates is shown to have little negative impact on the value of the method. Further effort may thus be fruitful to assist the requirements engineer in handling the large set of requirements found in a market-driven development organisation.

1.2. Related Work

The role of natural language processing in requirements engineering is discussed in Ryan [12], where the conclusion is drawn that natural language-processing techniques must be realistic and effort has to be made to identify where such techniques may be useful. It is argued that the validation of requirements still have to be an informal, social process. Thus, an automated system could or should not replace the human requirements engineer. Such systems are still not feasible or cost-effective to construct.

Various attempts have been made to use automated techniques to assist the analysis of requirements written in natural language:

1. Gervasi and Nuseibeh [13] use lightweight formal methods (low cost, partial analysis) to partially validate a syntactically correct NASA Software Requirements Specification (SRS) document. A glossary was manually produced from the SRS to aid the method.
2. Ambriola and Gervasi [14] present a web-based environment where Model–Action–Substitution rules and a domain- and system-specific glossary are used to extract abstractions and build models.
3. Rayson et al. [15] report on a project called

REVERE, where statistical and probabilistic natural language-processing methods are used to assist the analysis of complex and voluminous texts.

4. Park et al. [16] present a system that uses a sliding window model and a parser to support the analysis of requirements using a similarity measuring technique.
5. Rolland and Proix [17] present an environment that generates conceptual specifications from problem space descriptions written as sentences in natural language.
6. Osborne and MacNish [18] describe an approach to resolve ambiguities where only a controlled language is allowed when writing requirements in order to facilitate for a lexicon and grammar-enabled parser.
7. Cybulski and Reed [19] describe an elicitation method and a supporting management tool that help in analysing and refining requirements by using a parser, semantic networks, a domain-mapping thesaurus, and faceted classification schemes to allow proper formalisation of requirements written in natural language.
8. Chen et al. [20] present ideas where concepts in texts from electronic meetings are automatically classified by using automatic indexing, cluster analysis and hopfield net classification.
9. Landauer and Dumais [21] present the Latent Semantic Analysis (LSA) computational model for generation of a representation from large corpora. The representation captures the similarity of meanings of words and sets of words.

Although relevant and promising for several areas and approaches in requirements engineering, the above attempts do not address the situation described in the previous section. The main concerns in the context of this work are the following:

- Requirements are considered to be found in a separate document that is to be analysed, quality assured and produced before implementation begins. This is not the situation in the market-driven organisation where requirements arrive continuously and may, at any time, affect previous, current and coming releases of the software.
- The initial quality of the requirements is often considered to be adequate for semantic parsing. This may not be the case when requirements are collected from many different sources and stored in a database.
- Real industrial requirements are not always used to validate the methods or techniques presented. Accuracy and efficiency are not always reported.
- The semantic nature of invented requirements may not share the properties of regular corpora used in many linguistic approaches.
- Simple, robust methods can act as a baseline for better understanding and further improvements and comparisons of techniques.

Several approaches seem promising but we believe that more effort needs to be put into this field to reach consensus on which methods, techniques, approaches and tools may be appropriate for different types of developing organisations. In this paper we focus on the market-driven organisation and do not present a new model or a full-featured approach. Rather, the feasibility of using automated similarity analysis is empirically investigated using real industrial requirements and a benchmark is provided to which further effort may be compared.

1.3. Paper Structure

The paper is structured as follows. In Section 2 the situation of requirements similarity analysis in market-driven development is described. Section 3 explains how automated similarity analysis of natural language requirements may be performed. Section 4 presents a case study where actual requirements collected from industry have been analysed. The case study explores the quality of a simple automatic similarity analyser. In Section 5, further applications of automated support are presented together with a small study using the analyser from Section 3 to investigate if similar requirements also are interdependent. Section 6 identifies possible further work and improvements. In the final section the results are discussed and conclusions presented.

2. Requirements Similarity Analysis

Requirements carry information on which decisions are based. This information can be either *explicit* or *implicit*. The explicit information constitutes all the written text, drawn charts and other artefacts that are used as the basis for communicating requirements. The implicit information consists of all the assumptions, rules, standards and the domain knowledge possessed by the requirement issuers and the requirements analyst. When natural language requirements arrive at a rapid flow from many different issuers, a quick analysis is required to guarantee requirements' quality before they are used as a basis for further decisions. Although the linguistic quality of the requirements may be low it is often left unattended as the requirements make sense. Rather, the information explicitly stated may not give sufficient decision support. For this reason the requirements engineer uses implicit and explicit information accompanied by personal skills to analyse the requirements for completeness, ambiguity, similarity, etc. Completeness analysis is performed to ensure that enough information is included in the requirements to enable further refinement, such as setting priority, estimating effort and deriving new requirements (see example requirements in Fig. 4). Ambiguity analysis is performed to identify the risks of multiple interpretations among requirements. Similarity analysis is discussed below.

If supplementary information is needed to accept the requirement, the analyst may have to consult the issuer to make sure that the issuer and the analyst share the same interpretation. Thus, the requirements engineer acts to assure the quality of each requirement before allowing it to be further refined in the continuous requirements engineering process [6]. The situation is illustrated in Fig. 1, where example activities have been identified in the *quality gateway*.

The activities in the quality gateway are typically performed manually as there are few supportive tools available. The activities are tedious and time-consuming, but necessary in order to assure software quality and to satisfy market needs. It would therefore be highly beneficial if some of these tedious activities could be partly automated.

This paper focuses on similarity analysis, which is performed in order to find requirements that may be merged, grouped, eliminated or linked. For example, two similar requirements may be merged into one or may simply be grouped together to make sure they are handled simultaneously during development. A requirement may be similar to another to the extent that it is regarded as a duplicate and thus eliminated. Furthermore, two requirements may be similar in a certain aspect that establishes some kind of interrelationship (such as dependencies between requirements and requirement decompositions). The requirements engineer may also find it desirable to split large requirements into two or more requirements, which may become similar or related to each other and other requirements in the database.

When the requirements engineer decides whether two requirements are similar or not, it is with regard to the implications for further development. Of course these decisions are made by humans, but computer analysis of explicit information expressed in natural language may supply the requirements engineer with information regarding similarity to support these decisions.

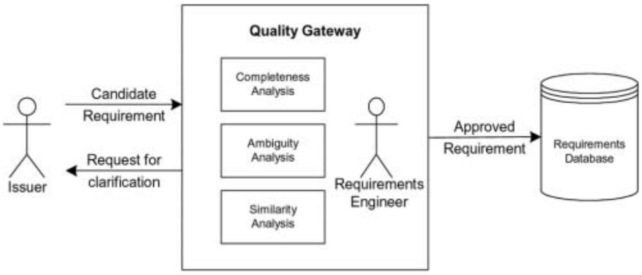


Fig. 1. Requirements quality gateway with three examples of quality assuring activities.

3. Automated Similarity Measurement

Statistical approaches to automated similarity measurement are widely used in information retrieval (IR), which is a well-established discipline concerned with automated storage and retrieval of documents written in natural language [22]. The presented work is based on existing IR techniques applicable in the analysis of natural language requirements. Figure 2 provides an overview of the steps in similarity measurement, where a similarity metric $S_{A,B}$ is calculated for a pair of textual requirements (A,B). The calculation of a similarity measure (further described in Section 3.1) is made subsequent to a number of pre-processing steps (elaborated in Section 3.2). The assessment of similarity metrics is described in Section 3.3.

3.1. Similarity Measures

In order to find relationships between requirements that may be merged, grouped or eliminated, a quantification of the degree of association between the requirements is needed. Several similarity measures are available, but no comparative studies exist that give a definite answer to which one to choose in this particular situation. In this paper we have therefore used three simple and well-known similarity measures to calculate the similarity between sentences: the Dice, Jaccard and cosine coefficients [23]. These measures all take the words in two sentences and calculate the similarity based on how many words they have in common. The coefficients are defined as follows, where A and B are requirements:

$$S^{D}{}_{A,B} = \frac{2\,|\{\text{words}_A \cap \text{words}_B\}|}{|\{\text{words}_A\}| + |\{\text{words}_B\}|}$$

$$S^{J}{}_{A,B} = \frac{|\{\text{words}_A \cap \text{words}_B\}|}{|\{\text{words}_A\}| + |\{\text{words}_B\}| - |\{\text{words}_A \cap \text{words}_B\}|}$$

$$S^{C}{}_{A,B} = \frac{|\{\text{words}_A \cap \text{words}_B\}|}{\sqrt{|\{\text{words}_A\}|\,|\{\text{words}_B\}|}}$$

All three measures have the desired property of normalisation, which imply that they give a value between 0 and 1 to indicate how similar a pair of

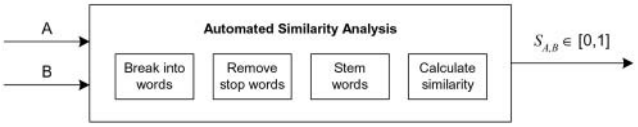


Fig. 2. A functional view of automated similarity analysis between requirement A and B, producing a measure $S_{A,B}$ ranging from 0 to 1.

sentences are, where 0 means that the sentences have no words in common and 1 means that the sentences are identical. The empirical investigation reported in Section 4 applies these measures to textual software requirements.

3.2. Preparing the Source Data

Before the similarity measure can be calculated the words of each sentence have to be extracted. This is achieved through *lexical analysis*, which takes an input stream of characters and converts it into a stream of words or tokens. This immediately raises the question of what should count as a word or token. For example, digits, hyphens, punctuation and letter case bring some problems that have to be considered. It is not technically difficult to solve these problems, but the chosen lexical analysis policy will affect the similarity measure. For example, preserving letter case will distinguish the words like ‘System’ and ‘SYSTEM’ and thus produce lower similarity measures. How to choose the policy thus depends on what type of data is to be analysed and the expected outcome.

Frequently occurring words like ‘a’, ‘the’, ‘of’, etc., will inadequately boost the similarity measures. These words, known as *stop words*, are therefore filtered out before similarity calculation. Which words to eliminate again depends on the type of data. It is reasonable to start out with a known stop word list that has been derived from general text.

Another issue is the *morphological variants* of words, i.e. the word forms. Words that are written in different forms usually carry the same information and should thus be considered equal. Therefore, words should be reduced to their ground form so that an automated word matcher would report a positive match. The technique used to reduce words to their ground form is called *stemming* and produces a stem from a word. For example, both the words ‘replace’ and ‘replacement’ may result in the stem ‘replac’ and consequently the words would be considered equal. There are several ways to stem words, such as affix removal, successor variety, table lookup, and n-gram [22]. In this paper we have used an affix removal stemmer, the Porter algorithm [24], which consists of a set of condition/action rules. It is a compact algorithm that has been shown to give good results in IR [22].

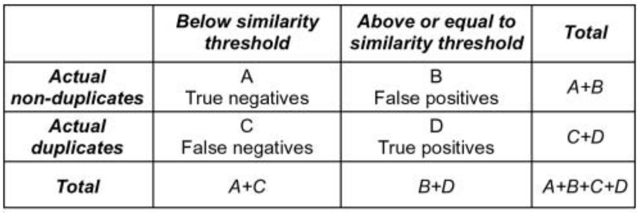
The similarity measure may be calculated by counting the number of stems produced from each requirement and the number of stems the requirements have in common. The common stems may be found using *exact match* or *inexact match*. Exact match requires the stem to be exactly equal, whereas inexact match calculates the similarity between the stems. Spelling errors may call for inexact match but brings the difficulty of choosing a good algorithm and a threshold level for match. The analyser used in this paper is designed to require an exact match between stems.

The low linguistic quality of the requirements will of course affect the similarity measure. However, we have chosen not to include spelling correction as we are interested in the performance of using a simple technique. It is also questionable if there is time for manual pre-processing in industrial settings.

3.3. Assessing the Quality of Similarity Measures

In order to evaluate the technique used to suggest similar requirements, a notion of quality is needed. We have chosen to use a contingency table, which defines a number of quality aspects in similarity measurement. Assume that $S(r_i, r_j)$ is a function that takes a pair of requirements and gives a similarity measure between 0 and 1. In addition we select a threshold value t, which acts as a selection criterion. If $S(r_i, r_j) \geqslant t$ then (r_i, r_j) is considered to be a suspected duplicate pair. Assume also that there exists a set of pairs of requirements that are identified as actual duplicate pairs. The similarity measure hence provides an approximation of this set of actual duplicate pairs, and the quality of the estimation may be defined according to Fig. 3 [23].

The resulting pairs that have a similarity value above or equal to the threshold level are regarded as duplicate pairs suggested by the analyser. Matches between actual duplicate pairs and those suggested by the analyser are



	Below similarity threshold	*Above or equal to similarity threshold*	*Total*
Actual non-duplicates	A True negatives	B False positives	*A+B*
Actual duplicates	C False negatives	D True positives	*C+D*
Total	*A+C*	*B+D*	*A+B+C+D*

True positives = $D/(C+D)$
False positives rate = $B/(A+B)$
Accuracy = $(A+D)/(A+B+C+D)$

Fig. 3. Assessment scheme with contingency table.

denoted *true positives*. The actual duplicate pairs not identified by the analyser are consequently denoted *false negative*s, i.e. they were wrongly suggested as non-duplicate pairs. The analyser may also suggest duplicate pairs that actually were non-duplicate pairs. These are denoted *false positives*. The rest are denoted *true negatives* and constitute all the requirement pairs that fell below the threshold level and were correctly suggested as non-duplicate pairs. The accuracy of the analyser is defined as the sum of the true negatives and the true positives divided by the total number of possible requirement pairs and indicates how well the actual duplicate pairs *and* non-duplicate pairs are identified. The total number of requirement pairs is calculated as $A + B + C + D$, which is equal to $(n \cdot (n - 1))/2$, when n is the number of requirements.

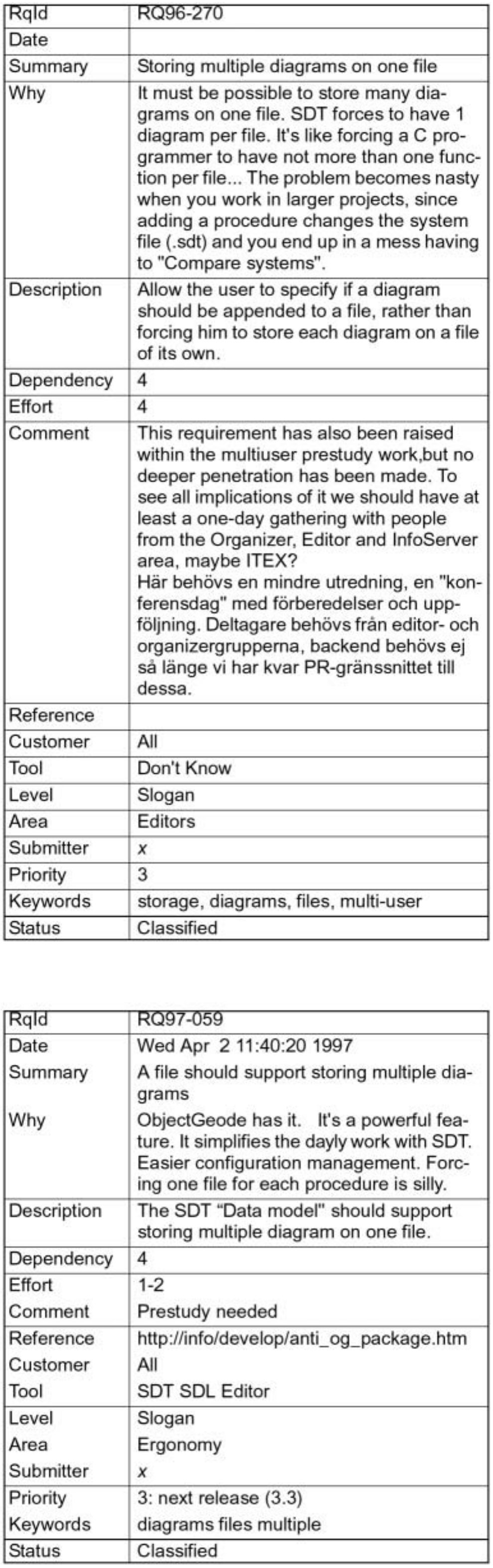
The contingency table will help reveal the performance of the method. In order to evaluate the feasibility of the analyser, a deeper investigation of the requirement pairs is needed. Taking any two identified pairs, they may or may not involve the same particular requirements. For example, the requirement pairs (A, F) and (C, F) share the requirement F. If the analyser assigns similarity values above zero to each of these pairs and a similarity value equal to zero to the pair (A, C) it would nevertheless be interesting to look at the three involved requirements together. We denote these preferred groupings of requirements as *n-clusters*, where n is the number of requirements in the cluster. The two single pairs in the previous example will thus form a 3-cluster. The cluster distribution can be derived by calculating the transitive closure of a graph in which the nodes correspond to requirements and edges correspond to pairs of requirements (r_i, r_j) with $S(r_i, r_j) \geqslant t$.

The sizes of the clusters and the number of clusters reveal the usefulness of the automated similarity analysis. It may be desirable to have many requirements grouped into n-clusters where n is the greatest number of requirements that the requirements analyst is capable of handling simultaneously. Example cluster distributions are presented in Fig. 6.

4. Empirical Investigation

In order to investigate the potential benefits of automated similarity analysis, we have applied the similarity measures described in Section 3.1 to real industrial requirements. The measures were used to see if automated analysis can correctly determine if a certain requirement is a duplicate of an already existing requirement.

For the investigation we have developed a computer program to perform the tasks specified in Fig. 2. The pre-



RqId	RQ96-270
Date	
Summary	Storing multiple diagrams on one file
Why	It must be possible to store many diagrams on one file. SDT forces to have 1 diagram per file. It's like forcing a C programmer to have not more than one function per file... The problem becomes nasty when you work in larger projects, since adding a procedure changes the system file (.sdt) and you end up in a mess having to "Compare systems".
Description	Allow the user to specify if a diagram should be appended to a file, rather than forcing him to store each diagram on a file of its own.
Dependency	4
Effort	4
Comment	This requirement has also been raised within the multiuser prestudy work,but no deeper penetration has been made. To see all implications of it we should have at least a one-day gathering with people from the Organizer, Editor and InfoServer area, maybe ITEX? Här behövs en mindre utredning, en "konferensdag" med förberedelser och uppföljning. Deltagare behövs från editor- och organizergrupperna, backend behövs ej så länge vi har kvar PR-gränssnittet till dessa.
Reference	
Customer	All
Tool	Don't Know
Level	Slogan
Area	Editors
Submitter	x
Priority	3
Keywords	storage, diagrams, files, multi-user
Status	Classified

RqId	RQ97-059
Date	Wed Apr 2 11:40:20 1997
Summary	A file should support storing multiple diagrams
Why	ObjectGeode has it. It's a powerful feature. It simplifies the dayly work with SDT. Easier configuration management. Forcing one file for each procedure is silly.
Description	The SDT "Data model" should support storing multiple diagram on one file.
Dependency	4
Effort	1-2
Comment	Prestudy needed
Reference	http://info/develop/anti_og_package.htm
Customer	All
Tool	SDT SDL Editor
Level	Slogan
Area	Ergonomy
Submitter	x
Priority	3: next release (3.3)
Keywords	diagrams files multiple
Status	Classified

Fig. 4. Two example requirements from the database denoted duplicates in the database. These two requirements were also suggested as duplicates by the similarity calculator at the 0.75 threshold level using the cosine similarity measure.

Table 1. Number of requirements in the database and in the different sets prepared for analysis

Status	Original	A_{full}	A_{reduced}
New	406	406	12
Assigned	428	428	31
Classified	601	601	601
Implemented	252	252	252
Rejected	103	103	103
Duplicates	130	101	90
Total	**1920**	**1891**	**1089**
Duplicate pairs	–	*142*	*124*

processing steps are handled by a lexical analyser, stop word remover and stemmer (explained in Section 3). The stop list remover excludes words with low discrimination value, and consists of 425 words derived from the Brown corpus [25]. For the stemming of words, the Porter algorithm is applied [24]. The similarity calculation produces a list of requirement pairs along with a value for each pair representing the similarity measure.

Telelogic, a large software developer, has allowed us restricted access to a requirements database of 1920 confidential requirements. Telelogic develops software development tools for a wide market and handles requirements arriving at a high rate from several different stakeholders (about three requirements a day [7]). The requirements are submitted through a web interface and thereafter managed by requirements engineers [4].

In Fig. 4, two examples of requirements from the database are shown. Many of the attributes are set at different stages in the requirements process, reflecting the refinement of the requirement from submitted to implemented or rejected [4]. The stage is represented by the 'Status' and the possible stages are shown in the leftmost column in Table 1. The table also shows, in the second column, the distribution of the 1920 requirements over the different stages.

4.1. Preparations

When a requirements engineer analyses a requirement, the requirement is checked on many different properties. Three related properties are (1) whether or not it is regarded as a duplicate of another requirement already in the database, (2) if it is possible to merge it with another requirement and (3) if it should be split into two or more requirements before further analysis. If a requirement has one of these properties, it is assigned the 'Duplicate' status and an appropriate action is taken. When a requirement is merged, all the information is added to the requirement it is merged with. When a requirement is split, the information is distributed over two or more new requirements. When a requirement is a pure duplicate (property 1 above), no further action is taken with the information.

As shown in Table 1, 130 of the 1920 requirements were either duplicates, merges or splits. In the analysis, only those that are 'true' duplicates are considered, since we know beforehand that merges and splits will match partially and thus bias the result. When these were removed, 101 requirements remained. The resulting set is shown in column 3 of Table 1 (set A_{full}).

Some of the 101 duplicates involved more than one requirement. This means that a requirement may be denoted a duplicate of two other requirements. To resolve this we parsed every identified duplicate and constructed a set of unique duplicate pairs. However, doing this creates a set of duplicate pairs that may be related (which addresses the discussion about clusters at the end of Section 3.3). Therefore, we calculated all these relations and created new duplicate pairs to denote the relation. For example, if requirement A initially was denoted a duplicate of requirements B and C, and requirement D was denoted a duplicate of requirement C, we would first create the duplicates pairs (A, B), (A, C) and (D, C). Then we would add the pairs (B, C), (A, D) and (B, D) to fully reflect all possible relations. This is acceptable since the duplicate relation is transitive. That is, if both A and D are duplicates of C, then A would also be a duplicate of D.

According to the requirements database manager, not all the requirements having status New or Assigned had been analysed for duplicates, and it was only certain that those having priority 1 had been analysed. Therefore, we considered removing all requirements with status 'New' or 'Assigned', not having priority 1. After doing this we noticed that some duplicate pairs referred to the removed requirements. Thus, we decided to analyse two sets: one with all requirements and one with the 'New' and 'Assigned' requirements with priority not equal to 1 removed. As the second set does not include all the requirements addressed in the duplicate pairs, those pairs were removed from the duplicates pair set. The resulting number of requirements and duplicate pairs are shown in column 4 in Table 1 (set A_{reduced}).

The textual information used to represent each requirement was collected from the 'Summary' field, which corresponds to a short requirement title, and the 'Description' field, which corresponds to a further explanation (see the examples in Fig. 4). As these fields were empty for a subset of the requirements, three different requirement sets were prepared from each of sets A_{full} and A_{reduced}. The first set comprised all the requirements that had a non-empty 'Summary' field. The second set comprised all the requirements that had a

Table 2. Final sets prepared for the analysis

Non-empty field	B_{full}		B_{reduced}	
	Requirements	Duplicate pairs	Requirements	Duplicate pairs
Summary	1830	142	1085	124
Description	1570	99	915	86
Summary or description	1887	142	1088	124

non-empty 'Description' field. The third set comprised all the requirements that had a non-empty 'Summary' field *or* a non-empty 'Description' field (NB. Not *exclusive or.* Requirements having a non-empty 'Summary' field *and* a non-empty 'Description' field were included in the last set). In the analysis of the sets using both fields, the two fields were treated as one. Table 2 shows the number of requirements in each of the sets after the requirements with the empty fields had been removed.

4.2. Results

The similarity calculator was run once for each of the prepared requirements sets to calculate the three similarity coefficients described in Section 3.1. The quality was assessed by producing contingency tables for nine different threshold levels as explained in Section 3.3. The threshold levels ranged from 0 to 1 with a 0.125 interval. All the possible combinations resulted in 162 contingency tables (3 measurements · 2 sets · 3 fields · 9 thresholds = 162 tables).

In Table 3, nine contingency tables are shown for the analysis on the 'Summary' field of set B_{full} using the cosine similarity measure. The number of possible unique pair-wise comparisons, which is the same as the total number of possible unique requirement pairs, is denoted $A + B + C + D$ in the contingency table in Fig. 3, and corresponds to the sum of each column in Table 3. The first row shows the number of correctly identified duplicate pairs and decreases as the threshold increases. Most requirement pairs are, correctly, considered as non-duplicate as shown in the second row. Their number increases with the threshold level. The third row shows how many duplicate pairs the analyser identified that actually were not identified as duplicate pairs by the experts. Finally, in the fourth row are all the actual duplicate pairs that the analyser did not find.

The number of false positives and negatives at threshold level 1 may raise some questions. There may be false negatives because requirements concerning exactly the same issue may be worded differently. The reasons that there may be false positives are several:

1. A requirement may be partially implemented and result in new requirements. The implemented requirement and the new requirements may then have the same information in some textual attributes. Since none of these requirements are marked as duplicates in the database the automatic analyser may produce a false positive.
2. The compared textual attributes may be wrong and misleading, not reflecting the actual meaning of the requirement.
3. Two requirements may be highly related and concern the same issue and have the same information in one textual attribute. Nevertheless, they do not have to be duplicates.
4. If all non-matching words in two requirements happen to be stop words, and thus eliminated before the similarity calculation, the reduced requirements may give a similarity measure of 1 but actually have different wordings.

The rate of true positives, the rate of false positives and the accuracy (see Section 3.3) were plotted to compare the measurements and to see which would generate the best result. In Fig. 5(a–d), four graphs are shown to support the conclusions on:

- which measurements may be considered the best;
- whether or not the requirements with status 'New' or 'Assigned' and not priority 1 should be ignored;

Table 3. Contingency table data for the summary field coefficient in set B_{full} using the cosine similarity measurement

	0+	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1
True positives (*D*)	114	114	105	80	62	47	42	31	30
True negatives (*A*)	1,578,213	1,578,581	1,628,049	1,666,093	1,670,881	1,672,945	1,673,247	1,673,341	1,673,349
False positives (*B*)	95,180	94,849	46,555	8,111	2,864	449	146	52	44
False negatives (*C*)	28	28	35	61	80	93	100	111	112

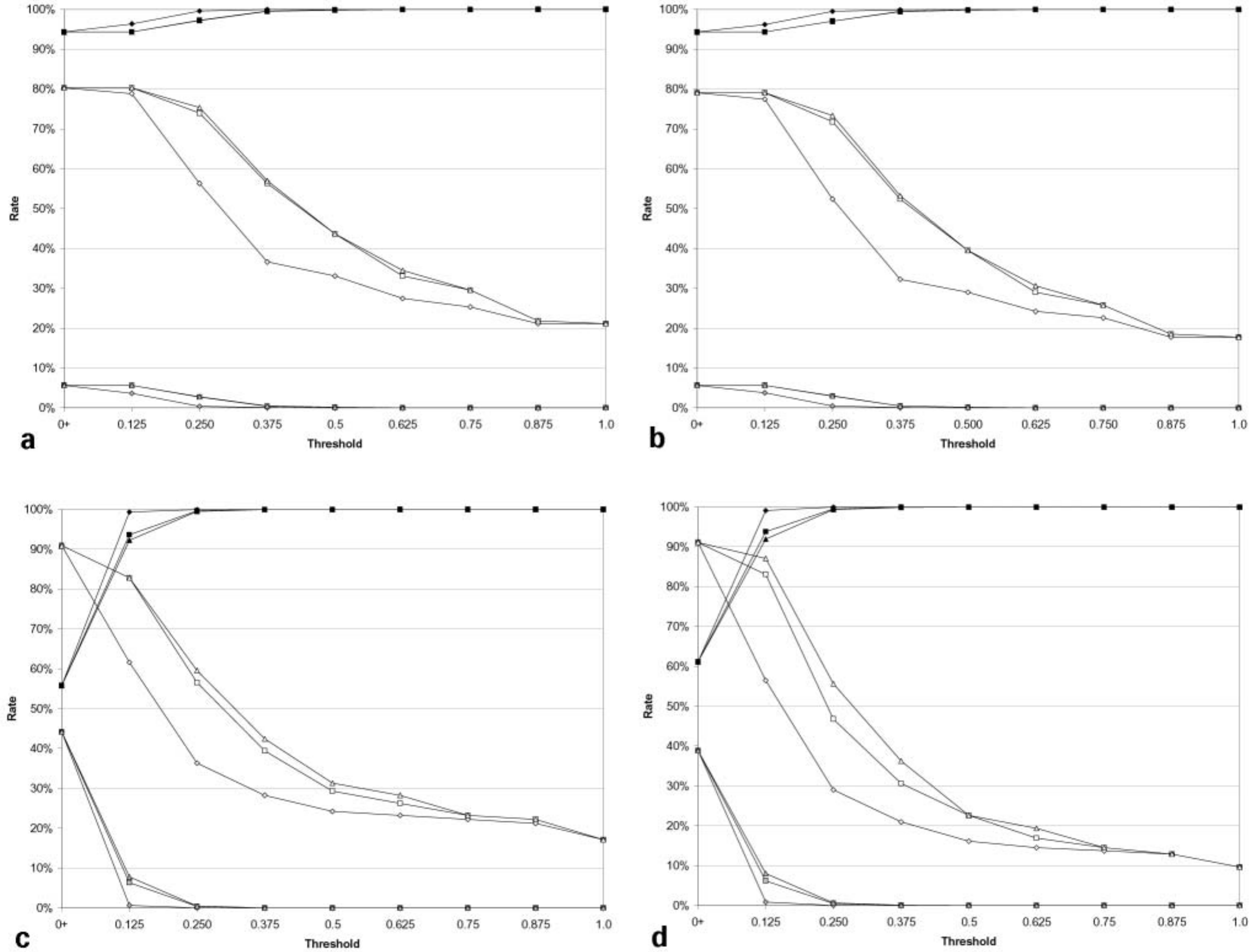


Fig. 5. (a) Similarity analysis performance using the summary field in set B_{full}. (b) Similarity analysis performance using the summary field in set B_{reduced}. (c) Similarity analysis performance using the description field in set B_{full}. (d) Similarity analysis performance using the summary and description fields in set B_{reduced}.

- which fields or combination of fields give the best results.

The graphs show that the rate of correctly identified duplicate pairs (the true positives rate) decreases from 80% or 90% at threshold level 0+ to about 20% at threshold level 1. The lowest degree of similarity is found when there is only one single word matching. Each measure will then give a similarity value just above 0 and thus, using threshold level 0+, suggest exactly the same set of duplicate pairs (every similarity measure but zero between two requirements results in a suggested duplicate pair). Correspondingly, the highest degree of similarity is found when all words match. Each measure will then give a similarity measure of 1 and produce exactly the same set of duplicate pairs. Between these threshold levels the curves differ slightly, which shows that the similarity measures perform differently. The Dice and cosine similarity coefficients show no significant difference, but the Jaccard coefficient performs slightly worse. Thus, for this particular set of requirements, the Dice or cosine coefficient is preferable.

The false positive rate is very low, decreasing from 5.69% down to 0.01%. The accuracy of the similarity analyser is as high as 94.3% at the lowest threshold level and increases to near 100% at threshold level 1. This curve suggests that the Jaccard coefficient is a better choice, contradicting the choice based on the positives rate.

Looking at the two topmost graphs, which show the results from using only the ‘Summary’ field, we can see that there is no considerable difference between the results for set B_{full} and B_{reduced}. This implies that either (1) there are ‘New’ and ‘Assigned’ requirements with lower priorities that have been analysed and found to be duplicates, of which some are identified by the program, or (2) the requirements have not been analysed and few matches were found by the program. Alternative 1 seems more plausible and is also confirmed by the contingency table – more duplicates are identified which must be related to the 'New' and 'Assigned' requirements with lower priorities.

The two leftmost graphs, showing the results from using the 'Summary' or the 'Description' fields respectively (from set B_{full}), differs on the low and high threshold levels. At threshold level 0+, the true positives rates is as high as above 90% using a combination of the 'Summary' and the 'Description' fields. However, the false positives rate is substantially higher and the true negatives rate has also dropped significantly. The conclusion from this comparison is that using only the 'Summary' field gives more accurate answers. The reason for this is that the 'Description' field contains too much noise that incorrectly boosts the similarity measures.

Finally, the top left and the two bottommost graphs support the rather evident: a combination of the 'Summary' and the 'Description' field results in a combination of the results from using the 'Summary' and the 'Description' fields separately.

The high number of requirement pairs identified at threshold level 0+ in Table 3 may at first seem very discouraging. However, calculating the cluster distribution of all the positives (true and false) as explained in Section 3.3 gives support to the following conclusions and the usefulness of the result.

The cluster distributions for the B_{reduced} set are shown in Fig. 6(a,b). Each figure shows four graphs. The first three show the cluster distribution using the cosine measure on the 'Summary' and the 'Summary' + 'Description' fields respectively. The last graph in each row shows the cluster distribution for the actual duplicates found by the experts.

The graphs show that with increasing threshold the number of clusters of larger size decreases. For example, in Fig. 6(a) at threshold level 0.375 there is one very large cluster involving 123 different requirements.

What is noteworthy about this is that the presented study is made on a very large set of requirements but that in reality the requirements arrive continuously, a few at a time. The similarity analysis can thus be made *incrementally* on a smaller set of requirements, avoiding the need for interpreting the results of similarity analysis of the entire set of requirements at one time. The cluster distribution shows that if we analyse one randomly selected requirement from the database (which may represent a newly submitted requirement), the worst case would be that the analyser suggests a cluster of 123 requirements to be identical (Fig. 6a, leftmost graph). This is thus the maximum number of requirements the requirement analyst must handle simultaneously. As the number may seem too high for the lower thresholds, it is reasonable to suggest that too large clusters may be ignored as they are probably irrelevant.

Considering both performance and cluster distribution, we may also conclude that the Dice and cosine measures are superior. The true positives rate has already been

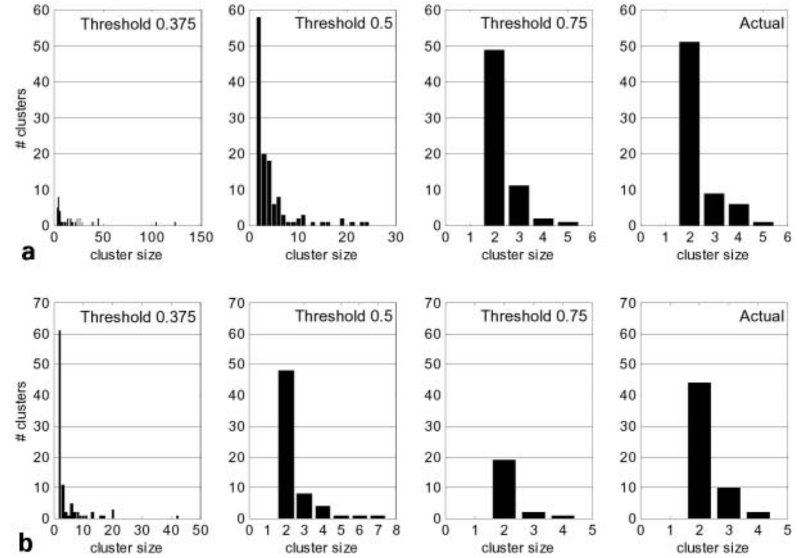


Fig. 6. (a) Requirements cluster distribution for the B_{reduced} set using the cosine measure on the 'Summary' field. The three leftmost graphs show the number of clusters of different sizes for various thresholds compared to the actual cluster distribution on the right. (b) Requirements cluster distribution for the B_{reduced} set using the cosine measure on the 'Summary' and the 'Description' fields. The three leftmost graphs show the number of clusters of different sizes for various thresholds compared to the actual cluster distribution on the right.

shown to be higher, and the higher false positives rate is compensated by the suggestion of analysing a group of related requirements simultaneously, instead of checking each of the several thousand possible duplicate pairs.

Another interesting issue is whether the automated analyser reveals duplicate pairs that the experts missed. To explore this we let an expert analyse the 75 false positives suggested when using the cosine measurement on the 'Summary' field for set B_{full} at threshold level 0.75. Table 4 shows the surprising result from the analysis. It turned out that 37% of the suggested duplicate pairs were actually missed by the experts! For that threshold level, the true positives rate would then increase from 26% (Fig. 5b) to almost 40%, the already low false positives rate would decrease, and the already high accuracy would increase. The analyst did not regard two requirements in a pair as duplicate or similar if they were to be implemented in different parts of the software. The table also shows the additional relationships identified, which thus imply that only 21 of the 75 pairs identified would be completely wrong.

The manual analysis also indicated that the analyser may have a problem when there are too few words in the fields. One suggestion would then be to use the 'Description' field only when the 'Summary' field has too few words.

Furthermore, the threshold value can be tuned based on the requirements engineer's consideration of the best trade-off between few false positives and many true positives.

In summary, it may be concluded that:

1. The similarity analysis technique gives reasonably high accuracy considering its simplicity.
2. For incremental analysis of requirements, given that related requirements are grouped into clusters, the Dice and cosine may be considered the superior measures.
3. A large explanatory field tends to give a worse result, as the discrimination between requirements declines. However, if one field has too few words it may be worth using other lengthy fields.
4. The grouping of suggested duplicate requirements into clusters reduces the analysis burden considerably.

Table 4. Result of expert analysis of the false positives for the set B_{full} at the threshold 0.75 using the cosine measure on the summary field

Relationship	Count
Duplicate	28
Similar	13
Related	8
Part of	5
Not related	21

5. Further Applications

There are numerous conceivable applications of automated similarity analysis beyond identifying duplicates. The following briefly describes some of these application areas, of which we have only evaluated one so far.

5.1. Requirements Interdependencies

Requirements interdependencies are important to identify and keep track of for requirements prioritisation and release planning purposes, as interdependencies may govern what partitions of a particular set of requirements are allowed from a functional perspective, or eligible from a cost/value perspective. Carlshamre et al. [6] describe a number of salient interdependencies found in a study of empirical data The relationship between similarity and interdependency is evident in the case where we have two requirements R1 and R2, with the exact same 'Sum-mary' field. This would be a true duplicate pair in the previous sense, but it would also represent an OR interdependency, which imply that either one of the requirements could be implemented. The existence of common keywords may indicate other types of interdependencies as well. For example, if there are several requirements that include the word 'sorting', it may be wise to consider implementing these together to save development resources, which would represent an interdependency regarding cost of implementation.

To investigate whether the similarity measurement technique could be used to support the identification of interdependencies in a set of requirements, we applied the same analysis technique as described in Section 3.1 to five different sets of 20 high-priority requirements, previously studied manually by experts (for further information on the results of the manual study, see Carlshamre et al. [26]). Among the total of 100 requirements, there were in total 155 pair-wise interdependencies manually identified by experts from each of the five organisations.

5.1.1. Results

Each set of 20 requirement slogans were relieved of stop words and reduced to stems, before being separately fed to the similarity calculator using the cosine coefficient. The automatic analyser reported 70 similar pairs on a 0+ threshold (9, 18, 21, 10 and 12 pairs in each set respectively), of which 25 were true positives. Table 5

Table 5. Contingency table for dependencies and similarities

	Similarity = 0	Similarity > 0	Total
Actual non-dependencies	750	45	*795*
Actual dependencies	130	25	*155*
Total	*880*	*70*	*950*

shows the frequencies of actual dependencies in relation to the similarity measure using the assessment scheme presented in Table 1.

A chi-square test [27] gives a p-value less than 0.0001, which shows that the similarity measure varies significantly with actual dependencies.

Thus, by checking for lexical similarity, this particular case demonstrates that it is a promising technique to support the interdependency identification process by automatic analysis. Although the accuracy may not suffice for this technique to be used on its own, automatic lexical analysis may be used in conjunction with other techniques to reduce the effort of identifying interdependencies.

5.2. Requirements Gathering

When a stakeholder is proposing a new requirement, it may be valuable to know if a similar requirement has already been implemented and, if so, in what release. If a similar requirement has not been implemented, it may be desirable to know if a similar requirement has been proposed.

5.3. Strategic Fit

A company may define key areas that are of specific importance for the requirements work (e.g., usability, decision-making features or invoicing capabilities). When such requirements are proposed, they can be identified by a similarity analysis approach and thus more easily be given the appropriate management attention.

5.4. Defect Tracking

Companies with mature software products that have gone through series of releases often have many defects to track and analyse. As new defects are reported, a similarity analysis approach can aid testers to identify if similar defects have been reported earlier.

5.5. Support Issues

Some companies allow their customers to get feedback on support issues through their web sites. Similarity analysis approaches can help the customer to enter questions in natural language and more easily analyse the questions and find suitable answers.

6. Further Improvements

There are a number of potential improvements that can be made to the presented requirements similarity measurement method, including the following suggestions to be evaluated in further research:

- Process issues such as when similarity analysis should be used, who should perform the analysis and how the analysis is cost-efficient to perform.
- How different ways of representing requirements affect the results. Which representation is best suited for high precision in automatic similarity analysis?
- Different attributes' impact on similarities. Use of other attributes may increase precision.
- Improve method accuracy. Examples include: the use of a domain-specific stop list, a thesaurus with general synonym words, spelling correction prior to the automated similarity analysis and by not discriminating between words with a short editing distance.
- Smart algorithms: some words may be over-represented in the set of false positives. Removing these words may improve the precision. This is an example of where it may be possible to make the algorithm self-adjustable based on human corrections.
- Evaluate linguistic methods that may provide more precise analysis of natural language requirements on a semantic level. This may include the use of ontologies or word nets.
- Ways of visualising the results from automated similarity analysis and supporting the requirements engineer in the navigation among related requirements.

In order to make these improvements and to make the methods more general it is of course desirable to apply the methods to other requirement sets from industry.

Also, it is of great interest to compare different approaches and combinations of approaches. The implementation cost and computational effort needed for statistical methods, linguistic methods and other computational models (such as the LSA approach [21]) is of much interest for applications aimed at market-driven organisations.

7. Conclusions

Automated similarity analysis is a promising technique for supporting requirements engineers to identify requirements duplicates and interdependencies. This conclusion is drawn on the basis of empirical studies on industrial requirements. Automated analysis is, in the particular cases of the presented investigations, able to identify as many as 80% of the actual duplicates and still only incorrectly classify about 6% of all the possible requirement pairs.

When using automated similarity analysis for interdependency identification, a significant correlation was found between similarity and interdependency. The results show a correct classification of 16% of the actual interdependencies.

We do not believe that the presented technique can replace human judgement, but our results suggest that automated similarity analysis on a syntactic level using information retrieval techniques may be effective in pinpointing true duplicates and interdependencies. Further studies are needed in order to increase the understanding of the benefits and limits of automated analysis of natural language requirements [12]. It is especially important to conduct further research in real situations, where new requirements are continuously arriving from multiple sources, and where requirements are analysed incrementally by a requirements engineer with domain expertise. In these investigations it is also of importance to consider the relationship between effort needed to put a method to work in a market-driven company and the efficiency of the method. Conducting real-world studies is a necessary means for valid assessments of the benefits and costs of decision support systems in a market-driven requirements engineering context.

Acknowledgements. This work is partly funded by the National Board of Industrial and Technical Development (NUTEK), Sweden, within the REMARKS project (Requirements Engineering for Market-Driven Software Development) grant 1K1P-97-09690. A previous version of this paper was published at the Seventh International Workshop on Requirements Engineering: Foundations for Software Quality (REFSQ'2001). We would like to direct warm thanks to Per Runeson, Martin Höst and Thomas Olsson, all at the Department of Communication Systems, Lund, for their valuable input and enthusiastic suggestions.

References

1. Sawyer P, Sommerville I, Kotonya G. Improving market-driven RE processes. In: Proceedings of the international conference on product focused software process improvement (PROFES'99), Oulu, Finland, June 1999
2. Lubars M, Potts C, Richer C. A review of the state of the practice in requirements modeling. In: Proceedings of the first IEEE symposium on requirements engineering (RE'93), San Diego, USA, January 1993
3. Deifel B. A process model for requirements engineering of CCOTS. In: Proceedings of the first international workshop on the requirements engineering process (REP'99), Florence, Italy, September 1999
4. Regnell B, Beremark P, Eklundh O. A market-driven requirements engineering process: results from an industrial process improvement programme. Requirements Eng 1998;3(2):121–129
5. Potts C. Invented requirements and imagined customers: requirements engineering for off-the-shelf software. In: Proceedings of the second IEEE international symposium on requirements engineering (RE'95), York, UK, March 1995
6. Carlshamre P, Regnell B. Requirements lifecycle management and release planning in market-driven requirements engineering processes. In: Proceedings of the second IEEE international workshop on the requirements engineering process (REP'2000), Greenwich, UK, September 2000
7. Höst M, Regnell B, Natt och Dag J, Nedstam, J, Nyberg C. Exploring bottlenecks in market-driven requirements management processes with discrete event simulations. In: Proceedings of the workshop on software process simulation and modeling (PROSIM'2000), London, UK, July 2000
8. Karlsson J, Ryan K. A cost–value approach for prioritizing requirements. IEEE Software 1997;14(5):67–74
9. Luhn HP. A statistical approach to mechanized encoding and searching of literary information. IBM J Res Devel 1957; 1(4):309–317
10. van Rijsbergen C J. Information retrieval, 2nd edn. Butterworths, London, 1979
11. Mitra M, Buckley C, Singhal A, Cardie C. An Analysis of statistical and syntactic phrases. In: Proceedings of the fifth international conference on computer-assisted information searching on the Internet (RIAO'97), Montreal, Canada, June 1997
12. Ryan K. The role of natural language in requirements engineering. In: Proceedings of the first IEEE international symposium on requirements engineering (RE'93), San Diego, USA, 1993
13. Gervasi V, Nuseibeh B. Lightweight validation of natural language requirements. In: Proceedings of the fourth IEEE international conference on requirements engineering (ICRE'2000), Schaumburg, USA, June 2000
14. Ambriola V, Gervasi V. Processing natural language requirements. In: Proceedings of the twelfth international conference on automated software engineering (ASE'97), Lake Tahoe, USA, November 1997
15. Rayson P, Emmet L, Garside R, Sawyer P. The REVERE project: experiments with the application of probabilistic NLP to systems engineering. In: Proceedings of the fifth international conference on applications of natural language to information systems (NLDB'2000), Versailles, France, June 2000
16. Park S, Kim H, Ko Y, Seo J. Implementation of an efficient requirements-analysis supporting system using similarity measure techniques. Inform Software Technol 2000;42(6):429–438
17. Rolland C, Proix C. A natural language approach for requirements engineering. In: Proceedings of the fourth international conference of advanced information systems engineering (CAiSE'92), Manchester, UK, May 1992
18. Osborne M, MacNish CK. Processing natural language software requirement specifications. In: Proceedings of the second IEEE international conference on requirements engineering (ICRE'96), Colorado Springs, USA, April 1996
19. Cybulski JL, Reed K. Computer assisted analysis and refinement of informal software requirements documents. In: Proceedings of the fifth Asia–Pacific software engineering conference (APSEC'98), Taipei, Taiwan, December 1998
20. Chen H, Hsu P, Orwig R, Hoopes L, Nunamaker JF. Automatic concept classification of text from electronic meetings. Commun ACM 1994;37(10):56–73

21. Landauer TK, Dumais ST. A solution to Plato's problem: the latent semantic analysis theory of the acquisition, induction, and representation of knowledge. Psychol Rev 1997;104:211–240
22. Frakes WB, Baeza-Yates R. Information retrieval: data structures and algorithms. Prentice-Hall, Englewood Cliffs, NJ, 1992
23. Salton G. Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley, Reading, MA, 1989
24. Porter MF. An algorithm for suffix stripping. Program 1980; 14(3):130–137
25. Francis WN, Kucera H. Frequency analysis of English usage. Hougton Mifflin, New York, 1982
26. Carlshamre P, Sandahl K, Lindvall M, Regnell B, Natt och Dag J. An industrial survey of requirements interdependencies in software product release planning. In: Proceedings of the fifth IEEE international symposium on requirements engineering (RE'01), Toronto, Canada, 2001
27. Siegel S, Castellan NJ. Nonparametric statistics for the behavioural sciences, 2nd edn. McGraw-Hill, New York, 1988