Published in the Proceedings of the 12th International Requirements Engineering Conference (RE2004).

Speeding up Requirements Management in a Product Software Company: Linking Customer Wishes to Product Requirements through Linguistic Engineering

Johan Natt och Dag
Lund University, Sweden
Johan.NattochDag@telecom.lth.se

Vincenzo Gervasi
University of Pisa, Italy
Gervasi@di.unipi.it

Sjaak Brinkkemper
Utrecht University, The Netherlands
S.Brinkkemper@cs.uu.nl

Björn Regnell
Lund University, Sweden
Bjorn.Regnell@telecom.lth.se

Abstract

Developing large complex software products aimed for a broad market involves a great flow of wishes and requirements. The former are elicited from customers while the latter are brought forth by the developing organization. These are preferably kept separated to preserve the different perspectives. The interrelationships should however be identified and maintained to enable well-founded decisions. Unfortunately, the current manual linkage is cumbersome, time-consuming, and error-prone.

This paper presents a pragmatic linguistic engineering approach to how statistical natural language processing may be used to support the manual linkage between customer wishes and product requirements by suggesting potential links. An evaluation with real requirements from industry is presented. It shows that in a realistic setting, automatic support could make linkage faster for at least 50% of the links. An estimation based on our evaluation also shows that considerable time savings are possible. The results, together with the identified enhancement, are promising for improving software quality and saving time in industrial requirements engineering.

1. Introduction

The success of market-driven companies developing software products essentially depends on three parameters: (1) when new releases reach the market, i.e. how well the market window is targeted, (2) what content they have, and (3) the associated cost for development [20**?**]. Profit and market share in combination with user satisfaction are the driving forces of requirements analysis, prioritization, and selection or rejection for implementation [12]. To find market opportunities and to keep customers satisfied, new requirements must be elicited continuously, and old requirements must be re-evaluated with respect to evolving market needs. This continuous elicitation puts a high pressure on the organization as requirements arrive from different sources and emerge in different projects [10]. Requirements management in a product software company bridges the market interaction (existing customers, prospects, analysts) with the product development planning (content, resource planning, timing). We therefore have to distinguish between two major groups of requirements: customer wishes and product requirements. Customer wishes are expressions of the perceived market need. These are naturally subject to frequent change; as the product changes, so does the market need. Customer wishes make up a vital and valuable source of information for decision-making. They also enable better communication with each customer by using the customer's own perspective. Product requirements are the requirements that the developing company believes are worth to pursue, stated from the developing company's perspective. These are also used as a basis for product release planning, as well as for feasibility studies and, if selected for implementation, to start actual software development. Customer wishes and product requirements often emerge independently of one another and for several reasons it is essential to keep them separated. For example, several customers may express their wishes slightly different and without referring to the software or business architecture. However, a product requirement addressing all the differently stated wishes may include additional information that is required for decision-making and development but that should not be communicated back to the customers. Naturally, there are multiple associations between these two groups of requirements, which must be found and maintained. The links between customer wishes and product requirements constitute a conclusive piece of information for requirements prioritization and release planning. As always, resources are limited and only a subset of the product requirements may be implemented in the next release of the product. The linkage process is cumbersome. Each time a new customer wish arrives, it is a difficult and time-consuming task to find those that may be related to the wide variety of product requirements. In current practice, this task is often accomplished using simple search facilities, with consequences on effort and missing links. A well thought-out hierarchical requirements organization may help (e.g., based on the software architecture), but as the product gets more complex, re-

quirements will not always fit nicely into such a structure. Moreover, evolution of the architecture, the product, and the company focus deteriorates the maintenance of the requirements hierarchies.

In this paper we investigate the possibility of giving automatic support to the manual work of requirements linking. This is carried out using a pragmatic linguistic engineering approach [8]. Our long-term objective is to engineer an integrated, supporting, and semi-automatic system that deals with natural language requirements and which satisfies the constraints in the particular industrial setting described. The most vital constraints are the cost-benefit of such a system and the varying textual quality of the requirements, to which a system must be less sensitive. We have selected a set of robust techniques, well known within statistical natural language processing [15], that we use to calculate similarities between requirements based on word occurrences. These similarities may be used to present, for a selected customer wish, a top list of product requirements that are candidates for linkage. We present an evaluation using real requirements and manually identified links, received from industry. It turns out that the selected techniques may properly support linkage in an industrial setting for up to 50% of the links. Improvements are suggested together with interesting alternatives and supplementary approaches. The paper is structured as follows. In Section 2 the case study environment is presented. Section 3 describes, in more detail, the requirements set used in our study. In Section 4 we further describe the envisioned new supported situation and, in particular, the selected techniques. This is followed in Section 5 by a presentation of our evaluation. A discussion of related work can be found in Section 6 which is followed by a survey of interesting further work in Section 7. Section 8 concludes the paper.

2. Requirements Management Case Study

Baan, now part of SSA Global, develops large complex applications aimed for enterprise resource planning (ERP), customer relationship management (CRM), supply chain management (SCM), product lifecycle management (PLM), and business intelligence (BI). The applications are designed and developed as a framework with separate functional components for different business functions and business process elements. By the end of year 2000, the framework consisted of 250 modules and 10,000 components, comprising around 4.5 MLOC. Between 1998 and 2002, the third author designed and introduced a new requirements management process due to the high complexity of their development situation:

- the comprehensive and vital domain knowledge,
- the large volume of requirements,
- their distributed development organization,
- and the complex dependencies of requirements.

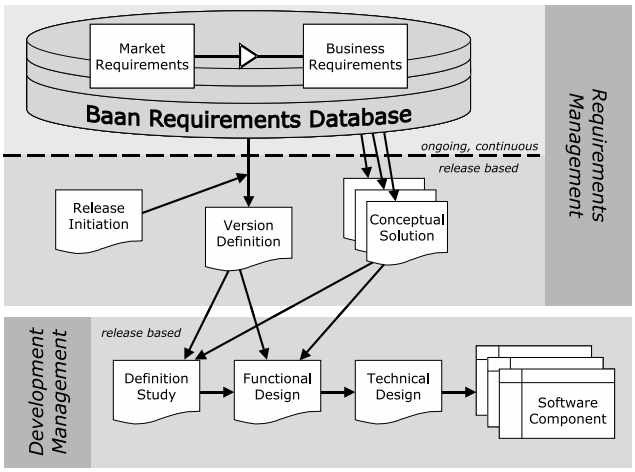

Figure 1. The Baan Requirements Management Process.

New requirements management concepts were introduced in order to make decision making more transparent and to enable more controlled requirements change management. To support the new requirements management process, Baan also developed their own requirements management tool called the Baan Requirements Database (BRD). Developed in MS Access, the BRD has successfully been used in a distributed setting (although, due to performance reasons, the system has recently been ported into the Baan-ERP development platform.). Obviously, the BRD is also used to collect requirements on the BRD itself. The requirements management process is depicted in Figure 1.

Requirements management is part of the overall release development process, which also consists of development management to develop the new releases, and delivery management to control the software component delivery to customers. The newly introduced concepts are as follows:

Market Requirements A customer wish related to current or future markets, defined using the perspective and context of the user. An example is found in Table 2.

Business Requirements A generic customer wish to be covered by Baan solutions described in Baan's perspective and context. An example is found in Table 3.

Release Initiation A formal document that triggers a release project in Baan containing high-level strategic topics for business requirement selection.

Version Definition A document with the listing of business requirements of the new release with the needed personnel resources.

Conceptual Solution A document with a sketch of the business solution for one (preferred) or more business requirements

As shown in Figure 1, the requirements management activities are executed in two modes. Continuously, new Mar-

ket Requirements (MR) and Business Requirements (BR) are being inserted into the BRD as soon as possible after their receipt or creation, respectively. Only after the company management decides to start a new release project, a Release Initiation document triggers the writing of the corresponding Version Definition (VD) and Conceptual Solutions (CS). Preferably, one CS covers one BR for the sake of simplified (de-)selection of BRs into the new release. The VD and the CS documents are then input for the development processes, which include the writing of design documents (Definition Study, Functional Design, Technical Design) and the coding of the software components.

MRs and BRs that cover the same underlying functional requirement are linked to each other. The relationship between MRs and BRs is essentially of many-to-many cardinality. MRs are copied into the BRD as-is, i.e. without altering the original text as specified by the customer. Maintaining a good relationship with customers is facilitated by providing timely feedback to the customer on their input for new product functionality. The customer receives an informative message after input review and after completion of the release. Therefore, in case several customers suggest the same functional extensions, then these are each recorded in separate MRs. These MRs are later linked to the same BR.

BRs should reflect a coherent well-defined extension of the product and are created by Product Managers responsible for (a part of) the product. A BR description includes the effort in man-days required for development. Experience with implementing Requirements Management in some product software companies learns us that transparent decision making during release planning requires the BRs to be of a similar workload size (i.e. for Baan between 20 to 80 mandays). Too many small requirements make the list of BRs in the VD too long and cumbersome to manage. Too large requirements do not provide adequate insight in the content of the next release, and hinder effective communication. As customers do not specify their MRs according to these guidelines, it may well be that an MR is very large and therefore linked to many different BRs. Non-coherent MRs dealing with dispersed functional areas are also linked to different BRs.

Table 1. Number of elicited and linked requirements.

	Business Requirements		Market Requirements	
Year	**Elicited**	**Linked**	**Elicited**	**Linked**
1996	0	0	183	113
1997	5	4	683	262
1998	275	169	1,579	388
1999	709	261	2,028	502
2000	669	167	1,270	397
2001	1,000	153	864	224
2002	1,121	340	1,695	514
Total	**3,779**	**1,094**	**8,302**	**2,400**

Linking MRs to BRs and the other way round is of a daily routine for the product managers. Each time a new MR is inserted into the BRD, it is first checked by searching whether there is one or more BRs that already include the specified functionality. This process is very time consuming, as the tool only allows text search in the requirement description. Similarly, when a new BR is created, the corresponding MRs need to be found in the BRD, since the objective is to satisfy as many customers as possible. Finding all MRs that are covered by the BR at hand is virtually impossible, because of the large number of MRs and due to the time-consuming understanding of MR content. Advanced automated assistance to the MR-BR linking can improve the quality of the requirements management process and save costly man-hours of the product managers.

3. Case Study Requirements Data

In this section we provide descriptive statistics on the requirements set used in our study. The total number of business and market requirements elicited at Baan between 1996 and 2002 is found in Table 1. These requirements constitute the basis for the calculations presented in the coming sections. Table 1 also presents the number of requirements that manually have been linked to one another. This information is used to evaluate the outcome of the automatic calculations. Also, the table shows that links between BRs and MRs may cross year boundaries. Figure 2 shows the distribution of how many MRs that are linked to each BR, and vice versa. A one-to-one relationship is obviously the most common. Furthermore, it shows that it is more common to link several MRs to one BR, as opposed to the other way around.

Examples of an MR and a linked BR are found in Table 2 and Table 3 respectively (some proprietary information has been left out). Although these two examples can not reveal the full picture, they are representative for the content and form of the two types of requirements. In the label and de-

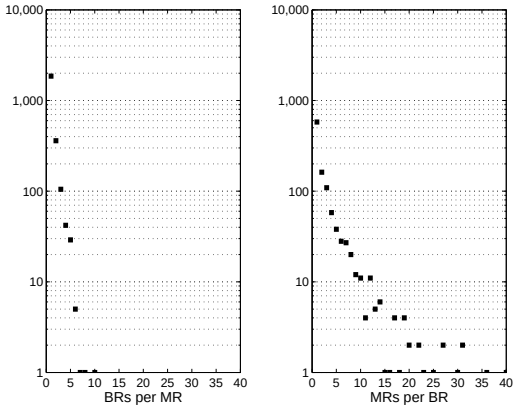

Figure 2. Number of linked requirements.

Table 2. Example market requirement.

Field	Example *[Description]*
Id	MR10739
Example	*[Request raiser's company]*
Request Person	*[Request raiser]*
Date	1996-05-29
Label	Pricing and Containerization
Description	Specifically what I am interested in is containerization and pricing. For a prospect I am working with (pretty much a distributor of electonic components) I need pricng by type of package by cusotmer type (wholesale or retail). I think pricing by container solves this problem, but I understand to use this feature the item must be a process item and I don't know if this is good or bad. If I must use process what do I gain or lose, like do I have to run a seperate MRP etc. Do I have to have one process company and one non-process company. They have mainly an assembly operation with no process involved. If process would be to cumbersome how difficut a mod would it be to disconnect containerzation from process.
Keywords	Pricing, order planning
Priority	Medium
Type	Functionality
Status	Closed/Completed
User name	*[Requirement submitter]*
Comments	020699: functionality is available in BaanERP in the Pricing module
Agreement	None

Table 3. Example business requirement.

Field	Example *[Description]*
Id	BR10025
Date	1998-01-27
Label	Statistics and containers
Description	1. Container (end item) in statistics Purchase and sales statistics used to be maintained only at main item level. But now it has also become possible to build statistics at container level. There are two aspects: printing statistics in the number of containers for a main item selecting and/or printing statistics at container level 2. Displays in statistics Displays are compositions of end items (for example, an attractive display of different types of cake). The statistics will be updated at both the levels of display item and container (which is part of the display). Prevention of duplicate counting, and correct pricing must be arranged in a procedural manner.
Keywords	Process industries
Type	Usability
Status	Assigned
User name	*[Requirement submitter]*
Comments	Warehousing only

scription fields we find the principal information that constitutes the requirement. The contents in these fields are written in natural language using the corporate language for documentation within Baan (US English). In the current situation, the association between the requirements would be found by, for example, searching for the term *container* in either of the corresponding sets of requirements. The dates of the requirements suggest that it is most likely that the MRs have been searched for possible linkage. Among the MRs we would get 37 hits if searching the label field and 318 hits if searching the description field. Five MRs were currently linked by experts (all five MRs were submitted earlier than the BR). Four links would be found through the label field, but the fifth link would only be found if selecting a new search term (e.g. *statistics*, 40 and 99 hits correspondingly). Based on this and similar cases, we expect the proposed technique to make this search and link procedure more efficient.

The requirements' textual quality varies, e.g. spelling errors (underlined), so in order to determine the requirements lexical and syntactic quality [4], we calculated term frequency statistics for different term categories. The term categories we were mainly interested in were correct words, misspellings, and abbreviations. Due to the very large number of terms used (see top row in Table 4), in order to reduce the manual effort needed for the quality assessment, we restricted ourselves to the subset consisting of only terms starting with 'a'. To further speed up the process, we first used WordNet 2.0 to automatically determine proper English terms. WordNet, developed at Princeton University, is a free lexical reference system in which English nouns, verbs, adjectives and adverbs have been organized into synonyms sets [6]. A manual check of the terms that were identified through WordNet was made to see if there were any terms that should be reclassified. We then manually classified the terms that were not found in the WordNet database. The results are shown in Table 4.

The left-most column shows the term categories that we identified. From top to bottom they are:

In WordNet 2.0 The terms that had an appropriate meaning in WordNet, i.e. an actual word, and excluding reclassified terms. For example, the term *au* was found in WordNet as (1) the chemical notation for gold, or (2) the abbreviation for astronomical unit. However, the term was not used in neither of these senses, but as a part of a web domain or as the French preposition. As English was the stipulated language we decided to reclassify the term as a reference.

Table 4. The requirements' lexical and syntactical quality (term beginning on letter 'a').

			Market Requirements				Business Requirements			
			Distinct		Frequency		Distinct		Frequency	
		Total number of terms:	27,239		659,325		10,431		334,059	
Letter A		**Total letter 'a' terms:**	1,247	(100%)	68,090	(100%)	793	(100%)	36,081	(100%)
		In WordNet 2.0:	662	(53%)	53,125	(78%)	538	(68%)	28,558	(79%)
		Actual words not in WN:	23	(1.8%)	12,675	(19%)	19	(2.4%)	6,679	(19%)
	'Non-words'	**Abbreviations:**	88	(7.1%)	1,009	(1.5%)	36	(4.5%)	584	(1.6%)
		Spelling Errors:	202	(16%)	284	(0.42%)	104	(13%)	124	(0.34%)
		Non-English:	142	(11%)	543	(0.80%)	14	(1.8%)	14	(0.039%)
		References:	67	(5.4%)	272	(0.40%)	62	(7.8%)	78	(0.22%)
		Persons:	36	(2.9%)	128	(0.19%)	4	(0.5%)	12	(0.033%)
		Merged:	22	(1.8%)	47	(0.069%)	15	(1.9%)	29	(0.080%)
		Code:	5	(0.40%)	7	(0.010%)	1	(0.13%)	3	(0.0083%)

Table 5. Most frequent terms.

BR rank	MR rank	Term	Frequency	$\frac{MRfreq.}{BRfreq.}$
1	1	item	2,117	2.30
2	2	line	1,609	1.83
3	6	process	1,484	1.50
4	15	datum	1,350	1.41
5	9	plan	1,281	1.66
6	136	erp	1,243	0.37
7	4	time	1,242	1.93
8	5	sale	1,137	2.09
9	7	change	1,078	2.03
10	40	warehouse	1,057	1.08
11	14	date	1,003	1.95
12	12	invoice	994	2.01
13	21	base	962	1.59
14	26	requirement	962	1.44
15	3	customer	956	2.72

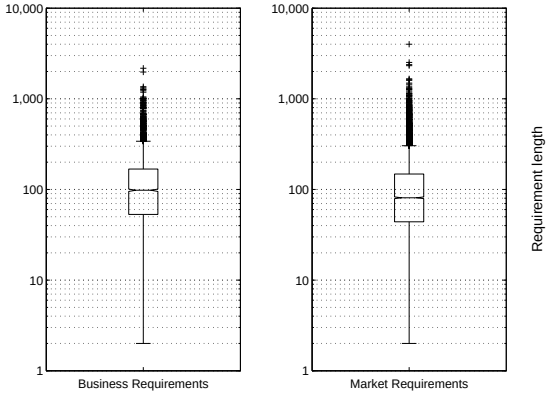

Figure 3. Requirements' length.

Actual words not in WN The terms that were not in WordNet, but manually identified as actual words. The numbers are not surprising, as WordNet only comprises four parts of speech and all senses are not represented in WordNet 2.0.

Abbreviations Non-standard abbreviations (e.g., accts for accounts) and other domain-specific abbreviations (e.g., ACP).

Spelling errors Misspelled words.

Non-English Words in another language than the stipulated. In most cases, in particular for the MRs, this is acceptable, as many non-English customers prefer to put an explanation in their own language within parentheses.

References Identifiers, names of companies, and product names (e.g. AOL, AG0000083).

Persons Names of people.

Merged Run-together words (e.g. articlegroup).

Code Pure programming code (e.g. AddShow).

As shown in the table, the terms in the 'non-word' categories are, fortunately, not used particularly frequently. We will utilize this fact in the next section.

Table 5 shows the 15 most frequent terms used in the BRs and the corresponding rank in an ordered term frequency list for the MRs. The table shows that many of the terms are used with comparable frequency in the BRs and MRs. This is also indicated by the Spearman rank order correlation coefficient r_s [24] (also see [13] for a discussion on statistics for corpora comparison). Calculated on the intersection of the two frequency lists, we get $r_s \approx 0,78$, significant at the $p < .00003$ level. The intersection constitute 4,660 terms in total. 1,899 terms only occurred in the BR frequency list and 8,234 terms only occurred in the MR frequency list. These unique terms had very low frequency in their corresponding frequency list and, more importantly, essentially comprise non-English words and misspellings. Thus the correlation coefficient do give a good indication of the shared term usage. This gives support for the technical approach chosen, to calculate similarity based on word occurrences, which is described in the next section.

The final statistical data that we present reveal the requirements' lengths. Figure 3 depicts the distribution of

the requirements' lengths for the BRs and MRs. The BRs are typically somewhat more verbose. In each set there are some requirements that comprise more than 500 words, but the most common requirement length is around 100 words. The reason for the enormous length of some requirements is that they contain complete mail conversations that discuss the requirement scope.

4. Technical Approach

The envisioned automated support to the manual work of requirements linkage should be well integrated into the BRD (Section 2), giving relevant suggestions on corresponding requirements when requested. Figure 4 illustrates where this automated support would be adopted into the requirements management process (Figure 1). It is important to understand that nothing is linked automatically. Based on the similarity calculations, *suggestions* are given to a human who may or may not assign the suggested links. Our expectation is that relevant suggestions will be provided faster this way than if a human would have to select several different search terms and, for each of these, search through the database.

The challenge is to suggest requirements that are potential candidates for linkage without the aid of any conceptual models, predesign or requirements models, as none are available at the time of submittal of a new requirement. Modeling the 12,000 requirements, even incrementally, prior to selecting only a small subset for implementation is simply not regarded as cost-beneficial. Approaches using natural language processing techniques in order to model, validate, and help understand requirements are available but are not directly applicable here (see Section 6 for a further discussion). These approaches may present interesting opportunities, but regarding the very large amount of requirements we start off by choosing a more pragmatic angle.

We first define the notion of a link in more technical terms. In Section 2, we defined a link between a customer wish and a product requirement as an indication that they refer to the same software functionality - or, in other words, that they express the same intent. So, two requirements should be linked if they have the same meaning, although expressed in a different style and language. Unfortunately, it is still an unsolved task to functionally represent meaning in a way that can successfully be used by automatic systems [15]. Thus, we cannot make use of such a system without human intervention. We therefore choose to recast the challenge into suggesting semantic similarity based on term occurrences. We assume that MRs and BRs refer to the same functionality if they use the same terms, i.e. the same terminology. In an RE context this may be a reasonable assumption, as in this case the language used tends to be more precise than in literary text, and moreover both customer wishes and product requirements refer to the same domain (that is characterized by the same basic language). This is supported by the term usage (of which an extract is presented in Table 5). Whether this assumption is valid is a matter of empirical validation, which this paper addresses. It should be noted however that this is the same assumption made in many text retrieval systems that have been in widespread use in other fields (e.g., medical literature, legal references, etc.).

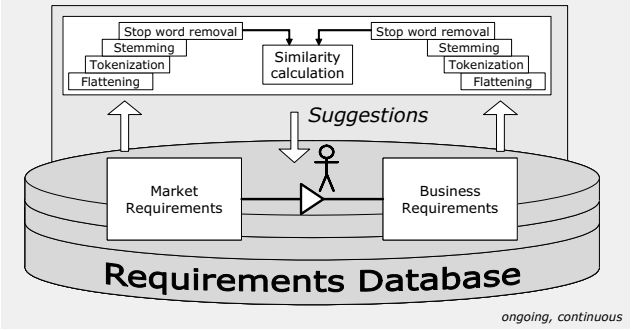

Figure 4. New setting with automated support.

4.1. Preprocessing

Before submitting the requirements to an automated process for establishing proper links, a number of preprocessing steps are performed. In detail, the process is as follows:

1. Requirements are first flattened, by merging the label and description fields, and discarding other administrative information (e.g., dates of submission). Thus, the complex data items maintained in the BRD are reduced to plain strings. This choice is based on the results in an earlier similar study [19], where the recall rate was found to be higher when using both a summary and a description field.

2. Each requirement is then tokenized, by using a custom-made tokenizer based on Flex/Lex rules [14]. In this step, terms are identified, and the original requirement (a string of characters) is transformed into a sequence of tokens. Particular care is taken to identify tokens that represent references to standards and other documentation (e.g., "ISO-8859-1"), as these occur frequently and are particularly meaningful for linking purposes. Other numeric references are left out altogether, as it turns out that they introduce too much unneeded noise in the data.

3. Stemming is then applied to each token to remove affixes and other lexical components not needed for comparison purposes. We use the `morpha` morphological analyzer described in [18] for stemming. For example, after this step both "managed" and "managing" are transformed into "manage", thus simplifying further processing.

Table 6. Intermediate results from preprocessing part of the example requirement MR10739.

Stage 1: Flattened	Stage 2: Tokenized	Stage 3: Stemmed	Stage 4: Stop words removed
Pricing and Containerization Specifically what I am interested in is containerization and pricing. For a prospect I am working with (pretty much a distributor of electonic components) I need pricng by type of package by cusotmer type (wholesale or retail).	pricing and containerization specifically what i am interested in is containerization and pricing for a prospect i am working with pretty much a distributor of electonic components i need pricng by type of package by cusotmer type wholesale or retail	price and containerization specifically what i be interest in be containerization and price for a prospect i be work with pretty much a distributor of electonic component i need pricng by type of package by cusotmer type wholesale or retail	price containerization specifically containerization price prospect pretty distributor electonic component pricng type package cusotmer type wholesale retail

Table 7. Vector-space matrix extract.

Term	MR10739	BR10025	BR10031
container	1	6	4
containerization	2		1
item	2	5	11
level		4	3
main		2	1
package	1		1
price	3	1	
print		2	
process	7		1
purchase		1	1
sale		1	1
sequence			1
statistics		8	
type	2	1	

4. Common terms that have a purely syntactic role (stop words) are then removed, as they do not provide useful information for establishing correct links. Articles, prepositions, and a few other closed-class words are discarded in this step.

As an example, Table 6 shows how a part of the MR in Table 2 is transformed into a sequence of terms by the preprocessing stages.

4.2. Linking and clustering

Each requirement, transformed accordingly, is then represented using a vector of terms with the respective number of occurrences (the so-called vector space model [15]). Table 7 presents an extract of the vector space matrix for the requirements set used in this paper. The matrix shows how many times a term appears in each requirement (notice that such a matrix is usually very sparse, and can be stored and queried efficiently). From the matrix it may also be derived how many terms the requirements have in common, i.e. the overlap. This can be used as an intuitive starting point for the similarity measure.

In the vector space model, each term can be seen as a dimension in an n-dimensional space. The number of occurrences of each term in a requirement is taken as the position of the requirement along the axis representing the term. Thus, a whole requirement can be represented as a point in the n-dimensional space. Very similar requirements will result in very closely clustered points in this space. Although a simple Euclidean distance measure could be used to identify similar requirements, better results can be obtained by using a measure that considers frequency of terms, rather than count of occurrences [15]. This is particularly true in our context, since often BRs are much more detailed and longer than the succinct customer wishes. The cosine correlation measure is often chosen in text retrieval applications for this purpose, as it does not depend on the relative size of the input [15]. In our case, we use the following measure:

$$\sigma(f,g) = \frac{\sum_t w_f(t) \cdot w_g(t)}{\sqrt{\sum_t w_f(t)^2 \cdot \sum_t w_g(t)^2}} \qquad (1)$$

where f and g are two requirements, t ranges over terms, and $w(t)$ denotes the weight of term t. The term weight is typically a function of the term frequency, since while the number of times a word occurs is relevant, its relevance decreases as the number gets larger [15]. One common approach is therefore to use a term weight of $1 + log_2(term\ frequency)$, which we use in this paper.

Once the similarity measure is defined, suggesting potential links for an incoming requirement is a matter of sorting pre-existing requirements according to their similarity to the new one, and offering the most-similar requirements to the user as candidates for establishing links. Of course, there is no guarantee that two requirements that are similar according to the $\sigma(\cdot)$ measure are indeed related: we assess how effective this technique is in industrial context in the next section.

At this point it is also worth noting at least two challenges, raised by the matrix extract, that the current preprocessing steps fail to handle. The stemming rules do not reduce the verb *containerization* and the noun *container* to the same stem. From a semantic point of view this is perfectly correct, but as the two terms concern the same domain concept their association should be utilized to increase the similarity measure. The current realization of the vector-space model will not make that possible.

The other potential problem has to do with synonyms (e.g. the term *purchase* would perhaps preferably be re-

lated to the term *buy*). Although synonyms may be relevant to address it is not certain that it will improve the measure considerably. I connection to synonyms it could be more promising to also take hypernyms and hyponyms into consideration [11]. Nevertheless, this is a matter for further research (see Section 7).

5. Evaluation

5.1. Evaluation technique

In order to evaluate how well the approach performs when it comes to identify correct links, we use the manually identified links as the ”presumably correct” answer. Our goal is to find out how many of these links the automatic system can retrieve. Retrieval results of this kind are traditionally evaluated by recall, precision, fallout, accuracy and error [15]. How these measures are to be calculated and interpreted is dependent on the application. Accuracy and error are often not very interesting as the number of correctly left out items (true negatives) usually is huge, which will give a high accuracy.

For the industrial setting described in Section 2 it is of interest to be presented, for a particular requirement of one type, with a list of candidate requirements of the other type. A top list is thus constructed by sorting the requirements by similarity. The size of the top list will thereby represent our similarity threshold.

A top list size of 1 is not necessary, nor wanted. A top list size of 7 ± 2 could be a good compromise [17]. It enables us to quickly spot one or more correctly related requirements, while taking into account that we are not able to reach 100% recall or precision (proper experimentation with presenting the resulting top list to the Product Managers is still required). In Table 8 the situation is illustrated, where an example extract of a top list for one MR is shown. In the table we have shaded those requirements grey that would fall outside the top list and which are typically not part of the shown result. The BRs in the top list that are correctly related to the MR are highlighted.

Table 8. Top list example

MR10013		
Pos	**Req.**	**Similarity**
1	BR10012	0.45
2	**BR10156**	0.43
3	BR10006	0.42
4	BR10536	0.38
5	**BR10987**	0.36
6	BR10273	0.36
7	BR10740	0.34
8	BR10419	0.33
9	**BR10622**	0.24
10	BR10082	0.21
11	BR10283	0.18
...	...	...

In this situation it is not critical that a correct suggestion is presented at position 1 but, of course, the higher the position the better. We could then use the ranked recall measure [11], but as we would like to relate the recall to a threshold (i.e. the top list size) we choose to compute recall for different top list sizes. Recall is the proportion of the target items that a system gets right (e.g. a system retrieving 100 of the 1,000 known relevant items has a recall of 10%) and we use the following adapted procedure:

1. Calculate the complete similarity matrix. The similarities are computed as described in Section 4.

2. For each requirement of one type, sort the requirements of the other type on similarity (as shown in the example in Table 8).

3. Calculate the overall recall for a top list of size n as:

$$Recall(n) = \frac{\sum_{i=1...\#req} targeted(n)}{\#actual\ links} \quad (2)$$

where

$$targeted(n) = \text{for requirement } i, \text{ number of correctly identified links within a top-}n\text{ list.} \quad (3)$$

We may then plot a recall curve as a function of the top list size.

However, step 3 is further adjusted to take the many-to-many relationships into account. Suppose we have the situation shown in Table 8. In the presented top list of 7 requirements, we find that 2 of them are correct. In an interactive situation, these may be marked for linkage and could then be removed from the top list. When they are removed, the requirements previously at position 8 and 9 may be revealed. In this example we are lucky, and the final third requirement is shown, which is easily spotted. As long as correct suggestions are shown in the top list we can reveal more suggestions without exceeding the selected top list size. Consequently, the recall rate may be slightly greater. This calculation procedure is more appropriate considering the specific industrial setting in which we expect to dynamically set the requirements relationships based on the presented top list.

5.2. Evaluation results

The results from our calculations are found in Figure 5. The figure shows the recall curve for the top lists of suggested BRs for each MR. The solid line represents the recall curve for calculating similarity using $1 + log_2(termfrequency)$, and the dashed line for calculating

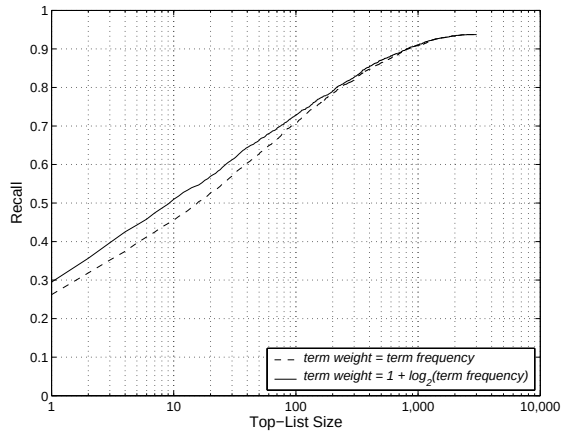

Figure 5. Recall for linking an MR to BRs.

it using just the term frequency. As expected, mitigation improves recall (typically 10%).

As can be seen we never reach 100% recall. This is because there are some links that could not be identified at all, i.e. some linked requirements have no terms in common. There were 204 links that could not be identified, which result in a maximum recall of $(3,259 - 204)/3,259 \approx 94\%$ (but that would require a top list of 3,000 requirements, which is quite unreasonable). Looking at the requirements comprised by the 204 links that could not be retrieved we found that:

- The links comprised 101 BRs and 158 MRs, thus including many-to-many links.
- The majority of the requirements were sparingly described, consisting of just a single line of text. In some cases there was no description at all. This is not necessarily wrong in the Baan RM process perspective (an empty BR is allowed to be created and directly linked to an MR). These special cases do however affect the results negatively.
- Some requirements were completely written in languages other than English. This should not be allowed without an additional English description.
- Many of the BRs and MRs seemed to describe completely different things. For a better understanding of these abstractions further analysis is required, which is beyond the scope of this paper. It is also of interest to understand how these links were assigned in the first place.

Figure 5 shows that, for a reasonable top list size of 10, we reach a recall of 51%. This is good considering the pragmatic approach taken and the impact on the saving of time that could be made in industry.

In an industrial setting, linkage is performed in both directions, i.e. MRs are searched to find which are related to a particular BR, and vice versa. We therefore calculated recall for the case where we are presented with a top list of suggested MRs for each BR. We can expect a difference, as the relative position of MR M within the top list for BR B, is not equal to the relative position of BR B within the top list for MR M (i.e., their individual positions are based on different similarity sets). We found that for a top list size of 10 we reach a recall of 41%. As we have no data on which direction was used originally to manually identify the links, we can only state that we reach a recall between 41% and 51% (for a top list size of 10). Notice however that we could also imagine a multi-page list of results, akin to the user interface of web search engines. In this case, we can expect the user to access up to two or three more pages of results if no convincing link is found in the first page, or if it is suspected that more requirements can be linked. In this case, in the most favorable setting with up to four pages we reach a recall close to 65%.

To give an indication of the time that could be saved, we make a rough estimate based on the statistics presented and another measure reflecting how many requirements that could be completely linked just by browsing a top-10 list. We found that for 690 of the BRs, the recall rate would be 100% using a top list size of 10, i.e. every related MR for each of the BRs would be found within a top-10 list. The 690 BRs are linked to 1,279 MRs, giving an average of 1.85 MRs per BR, but in order not to exaggerate the gain we assume that, in the manual case, one search term is enough to find all the links for one requirement. Supported by the search hit example in Section 3 we further assume that a search would return approximately 30 hits. Thus, in the manual case the worst case scenario would be to browse 30 requirements. With a top list size of 10, the worst case scenario with automated support would be to browse 10 requirements. Up to 66% effort could consequently be saved. If we assume that it takes about a quarter of a minute (15 seconds) to accept or reject a requirement as a link, we find that the gain is $(690 \cdot 30 \cdot 0.25) - (690 \cdot 10 \cdot 0.25)$ minutes $= 3,450$ minutes $= 57.5$ hours. The critical reader may protest that in a real setting it is not possible to know the stop criteria, i.e. how to know if a presented top-10 list comprise all the possible links for an arbitrary requirement. While that is true, it is also true in the current situation. The stop criteria will unfortunately always be unknown. The above calculation only gives a comparative evaluation of the effort that may be saved.

Finally, we also found it interesting to answer another question: how effectively can we find at least one correct link, i.e. for each one-to-many relationship how well is it possible to spot at least one of them? Just looking at the best-positioned suggested requirement, we reach a recall rate of 58% for a top list size of 10, which seems promising. This gives a reason to incorporate clustering techniques, i.e.

to cluster the requirements of one type and use that additional information when producing the top list.

5.3. Validity

There are four main validity threats that may adversely affect our results:

1. Correctness of the similarity calculations.
2. Completeness of actual links.
3. Degree of link intricacy.
4. Copy-and-paste of requirement content.

To address the first threat we have manually validated all the programs we developed. This was done for a randomly selected subset, for which we manually performed all the required steps and compared that to the automatically calculated results. Some minor bugs were found and corrected and for any changes to the code we regression tested all the programs.

The second threat has currently not been avoided. Working manually, Baan's product managers may have missed some relevant link, which our system has identified. Such a link would be considered incorrect in our evaluation. However, this threat is not problematic, since if the missing links are accounted for we will get higher recall. How much higher is beyond the scope of this paper (see the next section for a discussion on further work).

The third threat involves the difficulty of drawing the correct conclusion based on the kind of links that are among the correctly suggested ones. It may well be the case that the remaining links are much more difficult to find using the proposed techniques or not. Stated differently, the presented results may not be as promising as we may think. However, we do not claim to reach 100% accuracy and we do not aim at completely replacing the current practice. This threat should nevertheless be investigated further.

The final threat has to do with the fact that some BRs may have been created using the exact same text (or slightly modified) as in a specific MR to which it is then linked. Of course the system should spot these, but the recall curve may show a better result than is reasonable in an industrial setting. It is beyond the scope of this paper to manually analyze all 3,259 requirements links. A quick analysis was made to see if there were any requirements pairs that were assigned a similarity of 1. We conclude that although these were few (45), it is of interest to look specifically at those that have been assigned high similarity measures. It is a matter of further work to address this threat systematically.

6. Related work

A recent study shows that several software development companies, in particular the customer-oriented, use common natural language for specifying requirements in the early phases [16]. Due to the nature of the requirements management process in many companies, we believe that natural language will be used for several years ahead. This motivates the research efforts made within the field.

The underlying activity for supporting the linkage between BRs and MRs may be classified as *requirements similarity analysis*. The approach taken in this paper is based on the assumption that similar requirements have terms in common. This may certainly be an insufficient assumption for achieving very high recall rates and complementary approaches may offer improvements.

Similarity between textual requirements has not been studied extensively, but linguistic engineering techniques to aid other requirements engineering activities have been proposed that may present opportunities for improvements. In addition to our own previous work [19], there are a few that are related specifically to requirements similarity analysis.

For example, Goldin and Berry have developed a tool to extract abstractions from requirements sets [9]. The tool finds commonalities between requirements by using a sliding window technique that compares sentences character-by-character. They avoid some of the weaknesses in confidence and precision from using parsers or counting isolated words. The result from the tool by Goldin and Berry is a number of abstractions, selected based on the requirements' common content. Relating to our work, instead of extracting the abstractions, a similarity measure could be calculated based on this overlap.

A sliding window approach is also used by Park et al., this time on a word-by-word basis in order to index sentences [22]. They also use a parser to produce an alternative index. Similarity is then calculated for both sets and aggregated into a final, more accurate similarity measure. However, the requirements set used for the evaluation is small and larger sets may present more noise than is revealed in their evaluation. Nevertheless, their study shows how different techniques may be combined to improve the recall rate and this seems to be the most rewarding approach.

Other research efforts within requirements validation may also be incorporated to improve the similarity measures. This includes the work by Fabbrini et al. (a requirements quality model [5]), Cybulski and Reed (unifying the requirements terminology [2]), Rolland and Proix (conceptual modelling [23]), Fliedl, Kop and Mayr (conceptual predesign [7]), Osborne and MacNish (restricted language [21]), and Denger et al. (writing more precise requirements [3]). However, restrictions emerge from the specific setting described in Section 2. For example, it is not possible to force customers to write their requirements in a controlled language (proposed in [2, 21, 3]). As stated in the introduction and in Section 2, this is not even desirable. Furthermore, the problem addressed in this paper is not the difficulty of validating or understanding the requirements, but rather the challenge to handle the large amount of requirements in the decision phase prior to any software design efforts. Thus, it is too early to do any modelling (as presented in [23, 7]). Even if it was found to be valuable to model all

the 12,000 requirements, it would most likely require too much interactive manual labor.

In the end, it is a matter of the cost-benefit of the techniques to be used, not only what is virtually possible. The effort of getting support systems up and running and integrated into the requirements engineering process as well as making them perform good enough must be balanced against the benefit they provide.

7. Further work

The results presented in this paper are promising for further work and improvements. There are several issues that may have a positive impact on the recall curve:

- Incorporate and aggregate similarity measures using other available techniques (e.g. sliding window, part of speech tagging, etc.). For example, there are also arguments against the cosine measure (as it assumes Euclidian distance), which makes it interesting to investigate probabilistic measures.

- Reuse the information from already linked requirements. In a real setting, most requirements in the database would be already linked. Firstly, they could be used to get more accurate similarity measures, as more textual information would be available in the calculation. Secondly, they could in some cases be left out from the presented top lists. Thirdly, they could be used as a learning set: for each pair of terms (t, t') we could compute a bonus based on how many times the pair appears in pre-linked requirements (e.g., t in a MR, t' in a BR linked to the MR). When comparing to a new requirement, we could consider equal terms to match with a full score (e.g., 1.0), and different terms to match with their "bonus" (smaller) score. This way, a future occurrence of t would suggest that we should consider requirements containing t' as well.

- Expert validation. It is possible that not all links have been found in the manual work. Thus, it is possible that requirements at a high position in the resulting top lists actually should be linked. Experimentation with and interviewing of product managers about the missing links and the reasons (if any) for their rejection could lead to significant improvements.

- Incorporate semantics to catch more distant similarities. For example, tokenization and stemming could be replaced with a part of speech tagger (e.g. the Brill tagger [1]), compound concepts could be treated as terms, and WordNet or another lexicon could be used to deal with synonyms, hypernyms, and hyponyms.

To enable better matches it is also of interest to incorporate checking mechanism in the requirements submission stage, e.g. check of spelling (as implemented in Caliber RM) and language use. Such mechanisms would improve the results from the similarity calculations without complicating the technical design. The above issues should be addressed together with analysis of the requirement links that were assigned low similarity measures. That could reveal the nature of natural language requirements and how it would be possible to generically deal with them.

8. Conclusions

In this paper we have presented an approach to speed up requirements linking using Linguistic Engineering techniques. We have shown that for an industrial setting where numerous customer wishes and product requirements are elicited, there is valuable support to be given using already well-known, robust techniques.

We have shown that more than half of the links may be correctly suggested and thus found in an easier way than they are today. An estimation based on the evaluation also shows that for 63% of the linked product requirements, all links would be found within a ranked list of 10 suggested customer wishes and time savings of more than 65% could be made.

We argue that the linkage could be made quicker by pushing a button and select from a list of requirements, rather than choosing and typing different search terms. Even if the case is that only the easy 50% are found, it will still be easier and faster. Further investigations will reveal how simple or advanced links that may be found using further improved techniques.

A significant contribution of our work is that the approach has been evaluated using a large set of real industrial requirements. The varying quality that is always found in authentic requirements is a real challenge to natural language processing tools. Therefore, we believe it is of high importance to make these empirical validations before any further steps are taken.

Further technical improvements are as always possible, making way for important savings of time in industrial Requirements Engineering. We do not expect these savings to be of an order of magnitude, but if effort as indicated in Section 5.2 could be saved, we would call it considerable.

Linguistic Engineering techniques have not yet been fully exploited to support software product development. The challenge is to consider all the criteria to yield acceptance: usability, cost-benefit, flexibility, robustness and efficiency, to mention a few [8]. The presented results are promising for a step towards well-engineered systems to aid Requirements Management in companies that rely on communication in natural language.

Acknowledgments. The authors wish to thank Pierre Breuls and Wim van Rijswijk at Baan in Barneveld for kindly providing the requirements database. The Ernhold Lundström Foundation covered travel expenses for trips to Italy and the Netherlands. Per Runeson and Lena Karlsson provided valuable input, for which the authors are very grateful.

References

[1] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy, 1992.

[2] J. L. Cybulski and K. Reed. Automating requirements refinement with cross-domain requirements classification. In *Proceeding of the fourth Australian Conference on Requirements Engineering (ACRE'99)*, pages 131–145, Macquarie University, Sydney, Sep 1999.

[3] C. Denger, J. Dörr, and E. Kamsties. A survey on approaches for writing precise natural language requirements. Technical report, Fraunhofer Institut Experimentelles Software Engineering (IESE), Kaiserslautern, Germany, 2001.

[4] F. Fabbrini, M. Fusani, V. Gervasi, S. Gnesi, and S. Ruggieri. Achieving quality in natural language requirements. In *Proceedings of the 11th International Software Quality Week (QW'98)*, May 1998.

[5] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. The linguistic approach to the natural language requirements quality: Benefit of the use of an automatic tool. In *Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop*, pages 97–105, Greenbelt, Maryland, Nov 2001. IEEE.

[6] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[7] G. Fliedl, C. Kop, and H. C. Mayr. From scenarios to KCPM dynamic schemas: Aspects of automatic mapping. In *Proceedings of the 8th International Conference on Applications of Natural Language to Information Systems (NLDB 2003)*, pages 91–105, Burg (Spreewald), Germany, June 2003.

[8] R. Garigliano. JNLE Editorial. *Natural Language Engineering*, 1(1):1–7, Mar 1995.

[9] L. Goldin and D. M. Berry. AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation. *Automated Software Engineering*, 4(4):375–412, 1997.

[10] M. Höst, B. Regnell, and C. Wohlin. Using students as subjects – a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):201–214, Nov 2000.

[11] P. Jackson and I. Moulinier. *Natural Language Processing for Online Applications: Text Retreival, Extraction and Categorization*. John Benjamins, 2002.

[12] M. Keil and E. Carmel. Customer-developer links in software development. *Communications of the ACM*, 38(5):33–44, May 1995.

[13] A. Kilgarriff. Comparing corpora. *International Journal of Corpus Linguistics*, 6(1):97–133, November 2001.

[14] M. E. Lesk and E. Schmidt. Lex - a lexical analyzer generator. *Computer Science Technical Report*, 39, 1975.

[15] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 2002.

[16] L. Mich, M. Franch, and P. L. Novi Inverardi. Market research for requirements analysis using linguistic tools. *Requirements Engineering*, 9:40–56, 2004.

[17] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

[18] G. Minnen, J. Carroll, and D. Pearce. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223, Sep 2001.

[19] J. Natt och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. A feasibility study of automated natural language requirements analysis in market-driven development. *Requirements Engineering*, 7(1):20–33, 2002.

[20] R. J. Novorita and G. Grube. Benefits of structured requirements methods for market-based enterprises. In *Proceedings of the Sixth Annual International Symposium on Systems Engineering (INCOSE'96)*, Boston, MA, July 1996.

[21] M. Osborne and C. K. MacNish. Processing natural language software requirements specifications. In *Proceedings of the 2nd international Conference on Requirements Engineering (ICRE'96)*, pages 229–236, Colorado Springs, CO, 1996.

[22] S. Park, H. Kim, Y. Ko, and J. Seo. Implementation of an efficient requirements-analysis supporting system using similaity measure techniques. *Information and Software Technology*, 42(6):429–438, 2000.

[23] C. Rolland and C. Proix. A natural language approach for requirements engineering. In *Proceedings of the Fourth International Conference on Advanced Information Systems Engineering (CAISE'92)*, pages 257–277, Manchester, UK, May 1992.

[24] S. Siegel and N. J. Castellan, Jr. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, Singapore, 2nd international edition edition, 1988.