Using Artificial Intelligence Techniques to Enhance Traceability Links

André Di Thommazo[1,2], Rafael Rovina[1], Thiago Ribeiro[1], Guilherme Olivatto[1], Elis Hernandes[2], Vera Wernek[3] and Sandra Fabbri[2]

[1]*IFSP - São Paulo Federal Institute of Education, Science and Technology, São Carlos, SP, Brazil*
[2]*LaPES - Software Engineering Research Lab, Federal University of São Carlos, UFSCar, São Carlos, SP, Brazil*
[3]*Informatics and Computer Science Department, Rio de Janeiro State University (UERJ), Rio de Janeiro, RJ, Brazil*

Keywords: Requirements Management Techniques, Requirements Engineering, Software Engineering, Fuzzy Logic.

Abstract: One of the most commonly used ways to represent requirements traceability is the requirements traceability matrix (RTM). The difficulty of manually creating it motivates investigation into alternatives to generate it automatically. This article presents two approaches to automatically creating the RTM using artificial intelligence techniques: RTM-Fuzzy, based on fuzzy logic and RTM-N, based on neural networks. They combine two other approaches, one based on functional requirements entry data (RTM-E) and the other based on natural language processing (RTM-NLP). The RTMs were evaluated through an experimental study and the approaches were improved using a genetic algorithm and a decision tree. On average, the approaches that used fuzzy logic and neural networks to combine RTM-E and RTM-NLP had better results compared with RTM-E and RTM-NLP singly. The results show that artificial intelligence techniques can enhance effectiveness for determining the requirement's traceability links.

1 INTRODUCTION

Several authors highlight the importance of requirements management to the software development process (Zisman and Spanoudakis, 2004) (Cleland-Huang et al., 2012) (Sommerville, 2010). The majority of software errors found are derived from errors in the requirements elicitation and on keeping up with their evolution throughout the software development process (Salem, 2006).

Research performed by the Standish Group (1994) (2005) showed that the three most important factors to define whether a software project was successful or not are: user specification gaps, incomplete requirements, and constant changes in requirements. Notice that these factors are directly related to requirements management, which has the traceability feature as a key point.

One of the main elements to help the activities of requirements traceability is the requirements traceability matrix (RTM). The RTM is designed to register the existing relationships among system requirements. Due to its importance, it is the main focus of much research. Sundaram and others (2010), consider traceability determination and RTM to be essential in many software engineering activities, although it is a time consuming and error prone task. The authors claim that this task can be facilitated if computational support is given and the use of such automatic tools can significantly reduce effort and costs to elaborate and maintain requirements traceability and the RTM. These authors emphasize that such support is still very limited in existing tools. According to Cleland-Huang and others (2012), the research which has recently addressed requirements traceability has focused on the automatic traceability definition.

Among the ways to automate traceability, Wang and others (2009) highlight that current research makes use of three approaches:

- The spatial vector model (SVM): the work of Hayes and others (2003), this approach will be mentioned in Section 5.2.
- Semantic indexing: the work of Hayes and others. (2006) uses the ideas proposed by Deerwester and others (1990) from Latentic Semantic Indexing (LSI) in order to also automatically identify traceability. When LSI is in use, not only is the word frequency taken into consideration, but also the meaning and context

Di Thommazo A., Rovina R., Ribeiro T., Olivatto G., Hernandes E., Werneck V. and Fabbri S..
Using Artificial Intelligence Techniques to Enhance Traceability Links.
DOI: 10.5220/0004879600260038
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 26-38
ISBN: 978-989-758-028-4

used in its construction.

- The network probability model: the work of Baeza-Yates and others (1999) uses ProbIR (Probabilistic Information Retrieval) to create a matrix in which the dependencies among the terms are mapped in relation to the other document terms.

Given the aforementioned context, this article presents two approaches to automatically create the RTM using artificial intelligence techniques: RTM-Fuzzy based on fuzzy logic and RTM-N based on neural networks. Both of them combine two other approaches; RTM-E, that is based on functional requirements (FRs) entry data and RTM-NLP, that is based on natural language processing (NLP).

Fuzzy logic is being applied because as is known, traceability determination involves many uncertainties, and fuzzy logic has the ability to handle them. Besides, to improve the effectiveness of this approach genetic algorithms were used to determine the best configuration of the fuzzy system pertinence functions. Neural networks are being used because they can store knowledge acquired through examples and make inferences about new ones, using examples during the training phase. In our case, as we had executed some experimental studies, there was much data available to train the neural network, so motivating its use.

All these four approaches determine the level ("no dependence", "weak dependence" or "strong dependence") of the relationship between two FRs. To help the definition of the ranges that determine these levels generated by RTM-E and RTM-NLP a decision tree was used (Artero, 2009; Coppin, 2004) with the data obtained from previous experimental studies (Di Thommazo et al., 2012).

The four approaches were evaluated by a new experimental study to quantify the effectiveness of each one. It is worth mentioning that the RTM-E and RTM-NLP approaches had already provided satisfactory results in two previous experimental studies (Di Thommazo et al., 2012) (Di Thommazo et al., 2013). To make all these experiments feasible, the four RTM automatic generation approaches were implemented in the COCAR tool (Di Thommazo and others, 2007). COCAR tool supports some activities of Requirements Engineering. It is possible to store the description, processing, input data, output data, constraints and stakeholders of each FR of a system. The tool provides the generation of the RTM according to four approaches that are presented in this paper. Also, some reports on the FR can be generated.

This article is organized as follows: in Section 2 requirements management, traceability and RMT are introduced; Sections 3 and 4 present a brief definition of fuzzy logic and neural networks, respectively; in Section 5, the four RMT automatic generation approaches are presented and exemplified by using the COCAR tool; Section 6 shows the experimental study performed to evaluate the effectiveness of the approaches; conclusions and future work are discussed in Section 7.

2 REQUIREMENTS MANAGEMENT TECHNIQUES

Requirements management is an activity that should be performed throughout the whole development process, with the main objective of organizing and storing all requirements as well as managing any changes to them (Zisman and Spanoudakis, 2004) (Sommerville, 2010).

As requirements are constantly changing, managing them usually becomes a laborious and extensive task, thus making relevant the use of support tools to conduct it (Lai and Liu, 2009).

According to the Standish Group (2005), only 5% of all developed software makes use of any requirements management tool, which can partially explain the huge problems that software companies face when implementing effective requirements management and maintaining its traceability. Various authors emphasize the importance of tools for this purpose (Sommerville, 2010; Kannenberg and Saiedian, 2009; Goknil et al., 2011). Zisman and Spanoudakis (2004), for instance, consider the use of requirements management tools to be the only way for successful requirements management.

Two important concepts of requirements management are requirements traceability and traceability matrix, which are explained next.

2.1 Requirement Traceability

Requirements traceability concerns the ability to describe and monitor a requirement throughout its lifecycle (Guo et al., 2009). Such requirement control must cover all its existence from its source – when the requirement was identified, specified and validated – through to the project phase and implementation and ending at the product's test phase. Thus, traceability is a technique that allows the identification and visualization of the dependency relationship between one requirement and the others, and/or the other artifacts generated throughout the software's development process. The dependency concept does not mean, necessarily, a precedence relationship between requirements but, instead, how coupled they are to each other with respect to data, functionality, or any other perspective.

According to Guo and others (2009), requirements traceability is an important requirements management activity as it can provide the basis for requirements evolutionary changes, besides directly acting on the quality assurance of the software development process.

Zisman and Spanoudakis (2004) consider two kinds of traceability: horizontal traceability, when the requirements' relationship occurs between different artifacts like the requirements document (RD), models, source codes and, test artifacts; and vertical traceability, the focus of this paper, in which traceability is analyzed inside the same artifact, like the RD for instance. Through this artifact's FRs analysis it is possible to identify their relationship and generate the RTM.

2.2 Requirement Traceability Matrix—RTM

According to Goknil and others (2011), despite the various existing research treating traceability between requirements and other artifacts (horizontal traceability), only minor attention is given to the requirements relationship between themselves, that is, their vertical traceability. The authors also state that this relationship influences various activities within the software development process, such as requirements consistency verification and change management. A method of mapping such a relationship among requirements is RTM creation.

In addition, Cuddeback and others (2010) assert that an RTM supports many software engineering validation and verification activities, like change impact analysis, reverse engineering, reuse, and regression tests. In addition, they state that RTM generation is laborious and error prone, a fact that means, in general, it is not generated or updated.

Overall, the RTM is constructed as follows: each FR is represented in the i-th line and in the i-th column of the RTM, and the dependency between them is recorded in the cell corresponding to each FR intersection (Sommerville, 2010).

Guo and others (2009), Goknil and others (2011) and IBM (2012) have debated the importance and need of the RTM in the software development process, once such a matrix allows the prediction of the impact that a change (or the insertion of a new requirement) has on the system as a whole. Sommerville (2010) emphasizes the difficulty of obtaining such a matrix and goes further by proposing a way to subjectively indicate not only whether the requirements are dependent but how strong such a dependency is.

3 FUZZY LOGIC

Fuzzy logic was developed by Zadeh (1965) and, instead of simply using true or false, proposes the use of a variation of values between a completely false and an absolutely true statement.

In classic set theory there are only two pertinence possibilities for an element in relation to a set as a whole: the element pertains or does not pertain to a set (Artero, 2009). In fuzzy logic, pertinence is given by a function to which the real values pertain in a closed interval between 0 and 1. The process of converting a real number into its fuzzy representation is called "fuzzyfication". Another important concept in fuzzy logic is related to the rules that use linguistic variables in the execution of the decision support process. The linguistic variables are identified by names, have a variable content and assume linguistic values, which are the names of the fuzzy sets (Artero, 2009). In the context of this work, the linguistic variables are the traceability obtained by the three RTM generation approaches and may assume the values (nebulous sets) "no dependence", "weak dependence" or "strong dependence", which will be represented by a pertinence function.

4 NEURAL NETWORKS

Neural networks are inspired by the human brain and are composed of several artificial neurons. These neurons were created by McCulloch and Pitts (1943). In a neural network each neuron receives a number of input values. A function—called the activation function—is applied to these input values and the neuron activation level is generated as the function result that corresponds to the output value provided by the neuron. Neural networks are used to model complex relationships between inputs and outputs and have the ability to acquire knowledge for pattern recognition (Coppin, 2004).

There are multiple classifications for neural networks, depending on different characteristics:

- the number of layers or the type of connectivity:

fully connected or partially connected;

- the flow of the processed signals: feed-forward or feed-back;
- the way the training is done: supervised (when desired input and output data are presented to the neural network) or unsupervised (when only the input data are presented to the neural network and it is in charge of setting the output values). The training consists of presenting input patterns to the network such that it can adjust their weights. Thus, its outputs should present an adequate response when the input data provided are similar but not necessarily identical to those used in training (Artero, 2009).

An important kind of neural network is a multilayer perception (MLP) neural network. It is composed of source nodes that represent the network input layer, one or more intermediate layers, and an output layer. Except for the input layer, the others are composed of neurons. The MLP network connectivity is feed-forward; that is, the output of each neuron connects only to all the neurons of the next layer, without the presence of feed-back loops. Thus, the signal propagates in the network progressively. To develop the RTM-N approach an MLP neural network was used.

5 APPROACHES TO RTM GENERATION

The four RTM automatic generation approaches proposed in this work only take into consideration the software FRs and establish the relationship degree between each pair of them.

The RTM names were based on each taken approach. The approach called RTM-E had its traceability matrix named RTMe, the RTM-NLP matrix was called RTMnlp, the RTM-Fuzzy matrix was called RTMfuzzy, and the RTM-N matrix was called RTMn.

The approaches are implemented in the COCAR tool, which uses a template (Kawai, 2005) to store all requirements data. After the template is completed, the RD provides all the necessary data to evaluate the approaches. The main objective of such a template is to standardize the FR data, avoiding inconsistencies, omissions and ambiguities.

One of the template fields (which makes the RTM implementation feasible) is called Entry, and it is used to store the FR's data entry in a structured and organized way. It is worth mentioning here the work of Kannenberg and Saiedian (2009), which considers the use of a tool to automate the requirements recording task to be highly desirable. In the following, the approaches are presented.

5.1 RMT-E Approach

RTM generation is based on the FR input data. The dependency relationship between FRs is determined by the percentage of common data between FR pairs. This value is obtained through the Jaccard Index calculation (Real and Vargas, 1996), which compares the degree of similarity and/or diversity between the data sets of each pair. Equation 1 represents this calculation.

$$\mathrm{J(A,B)} = \frac{n(A \cap B)}{n(A \cup B)} \tag{1}$$

The equation numerator is given by the quantity of data intersecting both sets (A and B), and the denominator corresponds to the quantity associated to the union between those sets.

Considering FRa as the data set entries for a functional requirement A and FRb the data set entries for a functional requirement B, their dependency level can be calculated by Equation 2.

$$\mathrm{J(FRa, FRb)} = \frac{n(\mathrm{FRa} \cap \mathrm{FRb})}{n(\mathrm{FRa} \cup \mathrm{FRb})} \tag{2}$$

Each position (i,j) of the traceability matrix RTM(i,j) corresponds to values from Equation 3:

$$\mathrm{RTM}\,(i,j) = \mathrm{J(FRi, FRj)} \tag{3}$$

As COCAR stores the requirements data in an atomic way, once the data is inserted into a requirement data set, it becomes available to be inserted into the data set of another FR. This fact avoids data ambiguity and data inconsistency.

It is worth noting that initiatives using FR data entries to automatically determine the RTM were not found in the literature. Similar initiatives do exist to help determine traceability links between other artifacts, mainly models (for UML diagrams) and source codes, like those found in Cysneiros and Zisman (2008).

The determination of the dependency levels ("no dependence", "weak dependence" and "strong dependence") was initially carried out based on three RDs from applications of different domains. Such a process was performed in an interactive and iterative way, adjusting the values according to the detected traceability for each one of the three RDs. The levels obtained were: "no dependence" for values equal to 0%; "weak dependence" for values between 0% and 50%; and "strong dependence" for values above 50%. Adopting these intervals two experimental studies (Di Thommazo et al., 2012; Di Thommazo et al., 2013) were conducted. To improve the accuracy of the approach we decided to use the decision tree J48 (Coppin, 2004; Artero, 2009) aiming to detect the best intervals to define the levels of dependency. If we could be able to determine better intervals, the approach would be more effective, classifying correctly the relationship between two FRs. Decision tree is a technique able of finding the best intervals based on previous data (Artero, 2009). Hence, data from all RDs used in the previous experimental studies (Di Thommazo et al., 2012; Di Thommazo et al., 2013) were applied as input data to generate the decision tree.

As shown in Figure 1, based on the RDs, two types of matrix were generated: the RTMe and the RTMref, where this last was construed by specialists as will be detailed in Section 6. The percentage determined by the RTM-E approach (from 0 to 100% of dependence) and the correct level of each relationship link determined in the RTM-Ref were applied to create the decision tree. Hence, analyzing the decision tree new intervals were defined.

5.2 RMT-NLP Approach

RTM generation is based on NLP which, in the context of requirements engineering, does not aim for text comprehension itself but, instead, aims to extract embedded RD concepts (Deeptimahanti and Sanyal, 2011). There are many initiatives that make use of NLP to determine different traceability types in the software development process. However, few of them consider traceability inside the same artefact (Goknil et al., 2011). In addition, the proposals found in the literature do not use a requirements description template nor determine dependency levels as in this work. Hence, aiming to determine the dependency level between two FRs, the frequency vector and cosine similarity methods were used (Salton and Allan, 1994). This method determines a similarity percentage between two text excerpts.

To improve the process efficiency, text pre-processing is performed before applying the frequency vector and cosine similarity methods in order to eliminate all words that might be considered irrelevant, like conjunctions articles and prepositions (also called stopwords). For example: after this step the excerpt "*Allow warehouse users to check in/out a set of inventory items*", becomes "*Allow warehouse users check set inventory items*". After this, a process known as stemming is applied to reduce all words to their original radicals, levelling their weights in the text similarity determination. Thus the phrase that we are using as example becomes: "*Allow warehouse user check set inventory item*". After these two steps, the method calculates the similarity between two FRs texts: each excerpt is represented through a vector. The occurrences in each vector are counted to determine each word frequency. Both vectors are alphabetically reorded. Vectors have their terms searched for matches on the other and, when the search fails, the word is included in the "faulting" vector with 0 as its frequency. After this, it is applied the Equation 4 to calculate the similarity.

$$\text{sim(x,y)} = \frac{\sum_{i=1}^{n}(x_i \cdot y_i)}{\sqrt{\sum_{i=1}^{n}(x_i^2).\sum_{i=1}^{n}(y_i^2)}} \tag{4}$$

As in the RTM-E approach, the dependency level values had been defined according to the decision tree (J48): "no dependence" from 0% to 37%; "weak dependence" from 37% to 67%; and "strong dependence" for values above 67%.

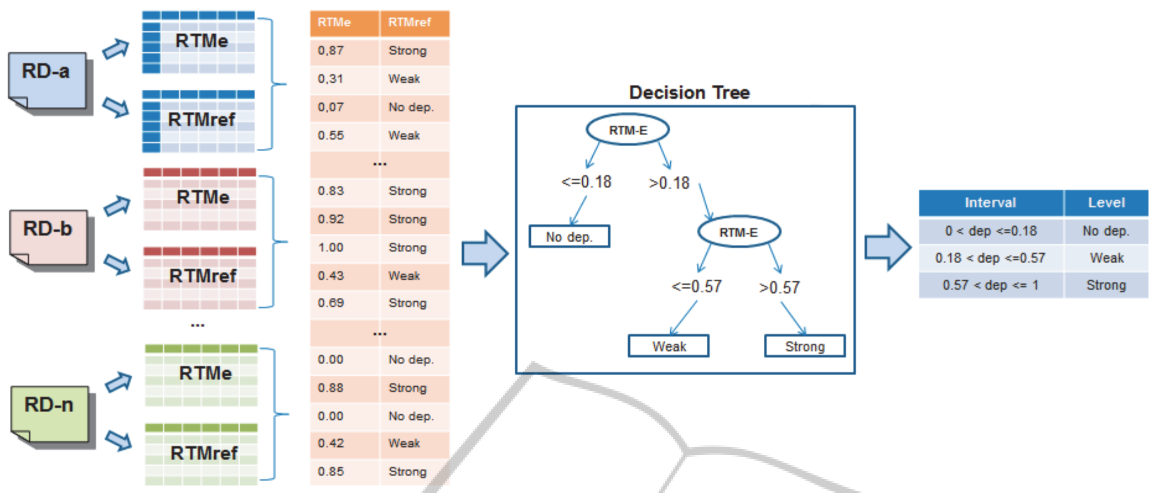
5.3 RMT-Fuzzy Approach

The RTM generation is based on fuzzy logic. The purpose of this approach is to combine, through a fuzzy system, the two approaches previously detailed. In this way, it is possible to consider both features—the relationship between the entry data manipulated by the FRs (RTM-E) and the text that describes the FRs (RTM-NLP)—to create the RTM.

Note that RTM-E and RTM-NLP determine the dependency levels ("no dependence", "weak dependence", and "strong dependence") between two FRs according to the value generated by the approaches. However, this method of calculating the dependency level can be very imprecise.

For instance, if the RTM-NLP approach generates a value of 56.5% for the dependency between two FRs, according to Figure 1, the dependency level would be "no dependence", whereas a value of 57.5% would indicate "weak dependence". Using the fuzzy logic, this problem is minimized due to the possibility of working with a nebulous level between those intervals through the use of a pertinence function, as mentioned in Section III. This conversion from absolute values to its fuzzy representation is called fuzzification.

In the pertinence functions, the X axis represents the dependency percentage between FRs (from 0% to 100%), and the Y axis represents the pertinence level, that is the probability of belonging to a certain fuzzy set ("no dependence", "weak dependence" or "strong dependence"), which can vary from 0 to 1.



RTMe	RTMref
0,87	Strong
0,31	Weak
0,07	No dep.
0.55	Weak
...	
0.83	Strong
0.92	Strong
1.00	Strong
0.43	Weak
0.69	Strong
...	
0.00	No dep.
0.88	Strong
0.00	No dep.
0.42	Weak
0.85	Strong

Interval	Level
0 < dep <=0.18	No dep.
0.18 < dep <=0.57	Weak
0.57 < dep <= 1	Strong

Figure 1: Steps to determine the new intervals level.

Figure 2 depicts the adopted fuzzy system, where RTM-E and RTM-NLP are the entry data. Table 1 indicates the rules created for the fuzzy system. Such rules are used to calculate the output value, that is, the RTMfuzzy determination. Initially, the values used for creating the pertinence functions were determined based on the experience of the authors of this paper. However, aiming to improve these functions, we used a genetic algorithm to reach better values for them.

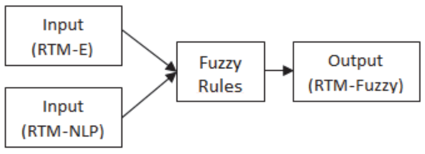


Figure 2: Fuzzy System.

The use of genetic algorithm to improve the results os Fuzzy System has been used in literature According with Herrera (2008) the automatic definition of an fuzzy rule based system can be supported by genetic algorithms.

Genetic algorithms are evolutionary algorithms which generate solutions to optimization problems. They are inspired by natural evolution and apply concepts like selection, mutation and crossover (Artero, 2009).

Another important concept is the concept of the chromosome, which corresponds to a set of properties of the "problem". Hence, the first step in using the genetic algorithm was to model the initial parameters of the pertinence functions in a chromosome. Figure 3 illustrates the chromosome definition scenario.

Table 1: Rules used in Fuzzy system.

	Antecedent		Consequent
if	RTM-E = "no dependence" AND RTM-NLP = "no dependence"	then	"no dependence"
if	RTM-E = "weak" AND RTM-NLP = "weak"	then	"weak"
if	RTM-E = "no dependence" AND RTM-NLP = "strong"	then	"weak"
if	RTM-E = "strong" AND RTM-NLP = "strong"	then	"strong"
if	RTM-E = "no dependence" AND RTM-NLP = "weak"	then	"no dependence"
if	RTM-E = "weak" AND RTM-NLP = "no dependence"	then	"weak"
if	RTM-E = "no dependence" AND RTM-NLP = "strong"	then	"weak"
if	RTM-E = "strong" AND RTM-NLP = "weak"	then	"strong"
if	RTM-E = "strong" AND RTM-NLP= "no dependence"	then	"weak"

Each function of each linguistic variable is used in the chromosome. As an example, considering the RTM-E approach defined by this author, data from the pertinence function "weak dependence" (Figure 3-A) was used to create the green part of the chromosome. After using the genetic algorithm the same pertinence function was modified as shown in Figure 3-B. The genetic algorithm process is summarized in Figure 4.

After the first chromosome creation (Figure 3), the next step is the initialization phase, when a generation of chromosomes is created. Thus, other chromosomes are randomly generated, with new values in some parts of the initial chromosome. In this case, a population of 100 individuals was used.

The next step is the selection phase where individuals that will continue in the next generation are selected. In this phase the roulette wheel selection algorithm was used, selecting 40% of the original population, based on the fitness function. This function evaluates the efficiency of each chromosome in solving the traceability detection.

To do this, we used data from a previous experimental study (Di Thommazo et al., 2012) since the values of RTM-E, RTM-NLP and reference RTM were known. The fitness function was used to summarize how close each chromosome was to the best solution.

After selecting the individuals that we will "keep alive" in the next generation, the genetic operators mutation and crossover were applied. When the mutation operator was applied a chromosome value (e.g. one cell of the green block) was randomly chosen to be changed with a new value (also randomly chosen). In the new population, 5% was generated through the mutation operator. The other individuals were generated using the crossover operator. Applying the crossover operator means that two chromosomes must be chosen to be crossed, generating new ones.

The process was iteratively executed until the termination phase was reached and the best chromosome was found. As the stop criterion we used 30 iterations. At the bottom of Figure 4 are the initial values of the first chromosome that generated the first population and the chromosome that reached the best results in the last population.

5.4 RMT-N Approach

RTM generation is based on neural networks. The purpose of this approach is to combine, through an MLP neural network, the two first approaches—RTM-E and RTM-NLP. An MLP neural network was used to develop the RTM-N approach. The process of training neural networks is detailed in Figure 5. Data from a previous experimental study (Di Thommazo et al., 2012) were used: the input data were the values of the RTM-E and RTM-NLP approaches and the output was the correct relationship, marked in the reference RTM (RTM-Ref). For example: considering the input values of 0.87 from RTM-E and 0.45 from RTM-NLP, it is necessary to set in the neural network that the output must be "strong dependence", since this is the value of the RTM-REF. This is achieved by setting the value "1" to the neuron that represents "strong dependence" and "0" to the two other neurons. After setting these values, the neural network should be trained, adjusting their weights to be able to detect similar inputs of RTM-E and RTM-NLP and correct results of dependence. If the input values were 0.42 from RTM-E and 0.58 from RTM-NLP, the neuron that represents the "weak dependence" must be set with "1" and the other with "0". The knowledge of patterns used to train the neural network came from the previous experimental studies, represented in Figure 5 by the DRs, RTMe, RTM-NLP and RTM-Ref.

Once the neural network has been created and trained, and it is provided with new input data obtained by the RTM-E and RTM-NLP approaches, the level of dependence between the involved FRs can be automatically identified (Di Thommazo et al., 2013).

To clarify the approaches it will be used the application of them a real system developed for a private aviation company (the full example is available at Di Thommazo et al., 2012 and Di Thommazo et al., 2013). To exemplify the RTM-E consider the following two FRs: FR3 related to products going in and out from a company's stock (warehouse) and FR5, related to an item transfer from one stock location to another. How they have some similar input data the RTM-E indicates a "strong dependence" (66% of common data) according to the Jaccard Index. The text of these two FRs also have a high similarity (88%), generating a "strong dependence" by RTM-NLP.

To exemplify the RTM-Fuzzy, consider the same FR3, previously mentioned, and FR7, related to the stock report generation. They do not have common entry data and, therefore, there is "no dependency" between them according to RTM-E. Despite this, RTM-NLP indicates a "strong dependency" (75.3%) between these FRs. This occurs because both FRs deal with the same set of data (although they do not have common entry data) and a similar scope, thus explaining their textual similarity. The fuzzy logic processing (presented in the Section 5.3), after applying Mandami's inference technique, generates the the value 42.5 for these entries, that corresponds to "weak dependence". To exemplify the RTM-N consider the same FR3 and FR5 already used in the first example. After the neural network was trained according to the process clarified in Section 5.4 it is ready to classify the FRs relationship. As the relationship between FR3 and FR5 generated by RTM-E was 66% and generated by RTM-NLP was 88%, these values are used as input to the neural network and the output indicates that the value of this relationship between FR3 and FR5 is "strong dependence".

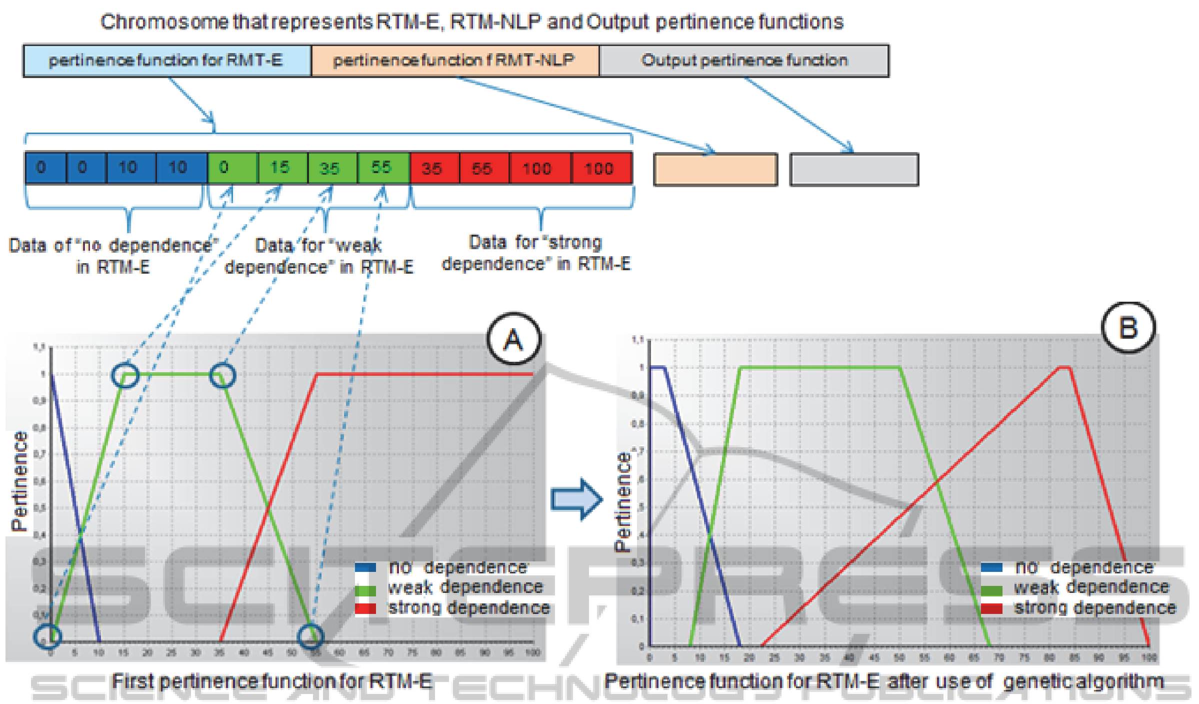

Chromosome that represents RTM-E, RTM-NLP and Output pertinence functions
pertinence function for RMT-E
pertinence function f RMT-NLP
Output pertinence function
0 0 10 10 0 15 35 55 35 55 100 100
Data of "no dependence" in RTM-E
Data for "weak dependence" in RTM-E
Data for "strong dependence" in RTM-E
A
B
Pertinence
no dependence
weak dependence
strong dependence
First pertinence function for RTM-E
Pertinence function for RTM-E after use of genetic algorithm

Figure 3: Creation of chromosome to be used in genetic algorithm.

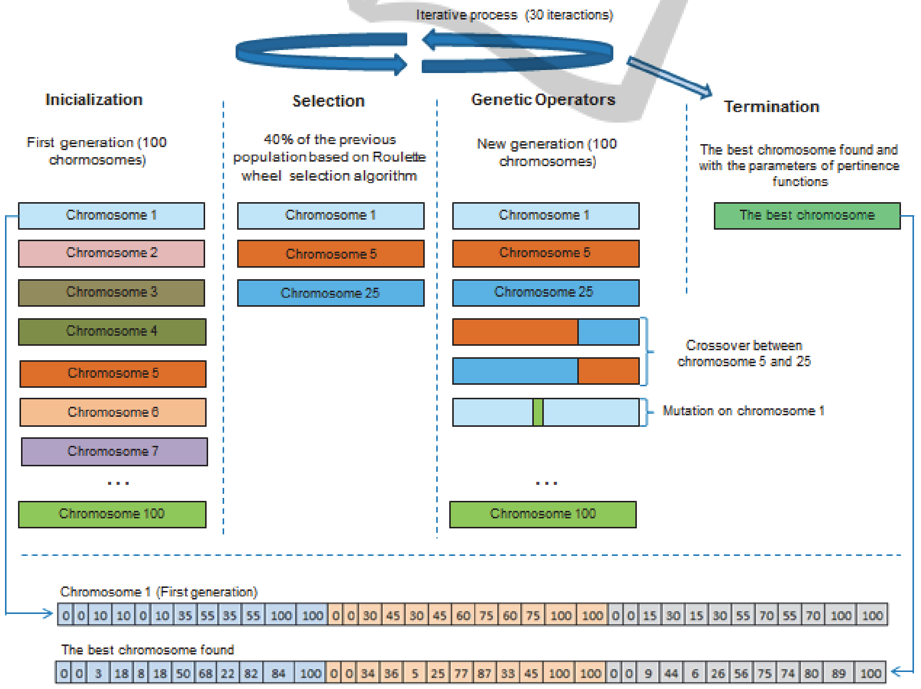
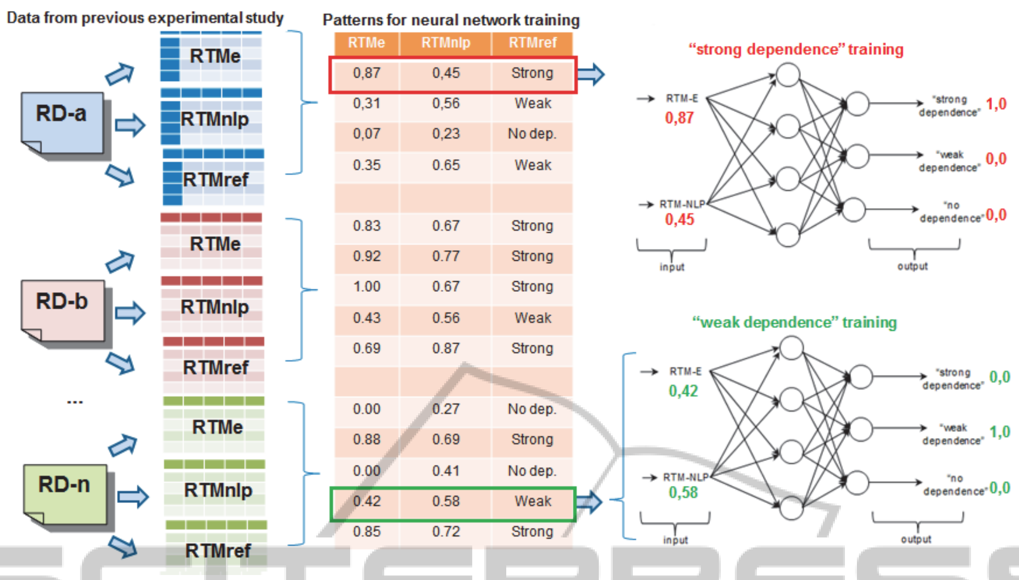


Figure 4: Steps of genetic algorithm.

6 EXPERIMENTAL STUDY

To evaluate the effectiveness of the proposed approaches, an experimental study has been conducted following the guidelines below:

Context: The experiment has been conducted in the context of the Software Engineering class at IFSP—Federal Institute of São Paulo—as a voluntary extra



RTMe	RTMnlp	RTMref
0,87	0,45	Strong
0,31	0,56	Weak
0,07	0,23	No dep.
0.35	0.65	Weak
0.83	0.67	Strong
0.92	0.77	Strong
1.00	0.67	Strong
0.43	0.56	Weak
0.69	0.87	Strong
0.00	0.27	No dep.
0.88	0.69	Strong
0.00	0.41	No dep.
0.42	0.58	Weak
0.85	0.72	Strong

Figure 5: Steps of training the neural networks.

activity. The experiment consisted of each pair of students conducting requirements gathering on a system involving a real stakeholder. The final RD had to be created in the COCAR tool.

Objective: Evaluate the effectiveness of the RTM-E, RTM-NLP, RTM-Fuzzy and RTM-N approaches in comparison to a reference RTM (called RTM-Ref) and constructed by the detailed analysis of the RD. The RTM-Ref creation is detailed next.

Participants: 32 undergraduate students on the Systems Development course at IFSP.

Artifacts utIlized: RD, with the following characteristics:

- produced by a pair of students on their own;
- related to a real application, with the participation of a stakeholder with broad experience in the application domain;
- related to the information systems domain with basic creation, retrieval, updating and deletion of data;
- inspected by a different pair of students in order to identify and eliminate possible defects;
- included in the COCAR tool after identified defects were removed.

RTM-Ref:

- created from RD input into the COCAR tool;
- built based on the detailed reading and analysis of each FR pair, determining the dependency between them as "no dependence", "weak dependence", or "strong dependence";
- recorded in a spreadsheet so that the RTM-Ref created beforehand could be compared to the RTMe, RTMnlp and RTMfuzzy for each system;
- built by the DR authors with supervision of this work's authors. The students were always in touch with the stakeholders whenever a reservation was found.

Metric: the metric used was the effectiveness of the three approaches with regard to the coincidental dependencies found by each approach in relation to the RTM-Ref. Effectiveness is calculated by the relation between the quantity of dependencies correctly found in each approach, against the total of all dependencies that can be found between the FRs. Considering a system consisting of *n* FRs, the total quantity of all possible dependencies (*T*) is given by Equation 5:

$$T = \frac{n(n-1)}{2} \qquad (5)$$

Therefore, the effectiveness rate is given by Equation 6:

$$\text{Effectiveness} = \frac{\text{Quantity of correct relationships found}}{T} \qquad (6)$$

Results: The results of the comparison between the data of RTMe, RTMnlp, RTMfuzzy and RTMn are presented in Table 2. The first column contains the name of the specified system; the second column contains the FR quantity; and the third provides the total number of possible dependencies between pairs of FRs. These values were calculated through Equation 5. The fourth, sixth, eighth and tenth columns contain the total number of coincidental dependencies between the respective approach and the RTMref. For example: if the RTM-Ref has

Table 2: Experimental study results.

			RTM-E		RTM-NLP		RTM-Fuzzy		RTM-N	
System	**Req Qty**	**# of possible dependencies**	**correct**	**effect.**	**correct**	**effect.**	**correct**	**effect.**	**correct**	**effect.**
Medical	17	136	115	85%	106	78%	124	91%	122	90%
Car Rental	23	253	204	81%	190	75%	215	85%	217	86%
Sales	18	153	122	80%	121	79%	130	85%	133	87%
Clothing Store	17	136	103	76%	105	77%	117	86%	119	88%
Cars	16	120	97	81%	93	78%	107	89%	107	89%
Habitation	16	120	103	86%	93	78%	110	92%	110	92%
Book Store	21	210	167	80%	157	75%	177	84%	180	86%
Pizza Delivery	16	120	95	79%	86	72%	104	87%	106	88%
Sales	22	231	195	84%	186	81%	213	92%	206	89%
Administration	17	136	103	76%	101	74%	113	83%	117	86%
Movies	17	136	103	76%	93	68%	112	82%	114	84%
Games	18	153	119	78%	118	77%	130	85%	131	86%
Food	19	171	134	78%	132	77%	144	84%	145	85%
Student	17	136	102	75%	94	69%	114	84%	117	86%
Computer Store	15	105	82	78%	78	74%	91	87%	91	87%
Ticket	20	190	159	84%	155	82%	169	89%	171	90%

determined a "strong dependence" in a cell and the RTM-E approach also defined the dependency as "strong dependence" in the same position, a correct relationship is determined. The fifth, seventh, ninth and eleventh columns represent the effectiveness of the RTMe, RTMnlp, RTMfuzzy and RTMn approaches, respectively, calculated by the relation between the quantity of correct dependencies found by the approach and the total number of dependencies that could be found (third column).

Results Analysis: Statistical analysis was been conducted using SigmaPlot software. Applying the Shapiro-Wilk test it could be verified that the data followed a normal distribution, and the results shown next are in the format: average ± standard deviation. To compare the effectiveness of the proposed approaches (RTM-E, RTM-NLP, RTM-Fuzzy and RTM-N) variance analysis (ANOVA) has been used for post-test repeated measurements using the Holm-Sidak method. The significance level adopted was 5%. The RTM-N approach was found to be the most effective with (87.3% ± 2.18), whereas the RTM-Fuzzy approach offered (86.6% ± 3.1) the RTM-E approach offered (79.6% ± 3.5), and the RTM-NLP obtained an effectiveness level of (75.7% ± 3.7). Based on these data it is possible to observe that the RTM-N and RTM-Fuzzy approaches (that combine the other two approaches through artificial intelligence) were more effectives on traceability detection than the RTM-E and RTM-NLP approaches singly.

In this experimental study, the results obtained by the RTM-E approach were similar to those obtained in two previous studies (Di Thommazo et al., 2012) (Di Thommazo et al., 2013), despite the fact that, in the first study, the RTM-NLP only presented an effectiveness level of 53%, which led us to analyze and modify this approach. In the second experimental study (Di Thommazo et al., 2013) this approach had an effectiveness level of 75%, very similar to that obtained in this experimental study. Even with such improvements, the approach still generates false positive cases, that is, non-existing dependencies between FRs. According to Sundaram and others (2010) the occurrence of false positives is an NLP characteristic, although this type of processing can easily retrieve the relationship between FRs.In the RTM-E data analysis, there were very few false positives. In most cases, the dependencies found, even the weak ones, did exist. The errors influencing this approach were due to relationships that should have been counted as "strong" being counted as "weak". As previously mentioned, if a relation was found to be "strong" in RTM-Ref and the proposed approach indicated that the relation was "weak", an error in the experiment's traceability was counted. In the case of relationships indicating only "dependence" or "no dependence", that is, without using the "no dependence", "weak dependence" or "strong dependence" labels, the effectiveness determined would be higher. In such a case the precision and recall metrics could be used, given that such metrics only take into account the fact that a dependency exists and not its level ("weak" or "strong") (Cleland-Huang et al., 2012).

In relation to the RTM-Fuzzy approach, the results generated by it were always higher than the results found by the RTM-E and RTM-NLP approaches alone.

A previous study with RTM-Fuzzy had an effectiveness level of 83%. To improve this result in this study, the approach was improved with the use of genetic algorithms, as detailed in Section 5. With the new fuzzy system pertinence functions, better results were found (86%).

In relation to the RTM-N approach the generated results were also always higher than the results found by the RTM-E and RTM-NLP approaches alone.

The experimental study herein presented has some threats to its validity. One of them is related to the students' inexperience in eliciting the requirements from the stakeholders. In an attempt to minimize this risk, known domain systems were used as well as RD inspection activities. The inspection was conducted based on a defect taxonomy commonly adopted in this context and which considers inconsistencies, omissions, and ambiguities, among others. Another risk is related to the correctness of the RTM-Ref. To minimize errors in this artifact the authors of this paper helped the students with this task and the stakeholder was contacted whenever there was any doubt about the relationship.

6 CONCLUSIONS AND FUTURE WORK

This article presents two approaches to automatically create the RTM using artificial intelligence techniques. RTM-Fuzzy is based on fuzzy logic and RTM-N is based on neural networks. They combine two other approaches: RTM-E, which is based on the percentage of entry data that two FRs have in common, and RTM-NLP, which uses NLP to determine the level of dependency between pairs of FRs.

Fuzzy logic was used to treat the uncertainties that might negatively interfere in the requirements traceability determination. Thus, RTM-Fuzzy uses the results presented in two other approaches but adds a diffuse treatment in order to perform more flexible RTM generation. Hence, RTM determination is a difficult task, even for specialists, and using the uncertainties treatment provided by fuzzy logic has been shown to be a good solution for automatically determining traceability links with enhanced effectiveness. To improve this approach, genetic algorithms were used to determine the pertinence function. Compared with a previous experimental study, after the use of this technique, the effectiveness improved from 83% to 86%.

Neural networks were used due to the possibility of using the knowledge from previous experimental studies conducted to detect the traceability links with RTM-E and RTM-NLP. With the data from these experimental studies a neural network was trained such that it was able to detect the traceability automatically. The results of the experimental study show that combining the other approaches through a neural network is a promising solution to automatically create the RTM.

From the four approaches here presented, it is worth noting that there are already some reported proposals in the literature using NLP for traceability link determination, mainly involving different artifacts (requirements and models, models and source code, or requirements and test cases). Such a situation is not found in RTM-E, for which no similar attempt was found in the literature. Comparing the four approaches presented in this paper with the others initiatives in the literature it is possible to say that, while the other approaches are limited to establish if "there is" or "there is no" dependence between two FRs, the approaches presented in this paper determine the level of dependence: "no dependence", "strong" or "weak". This feature allows a development team give a special attention to the "strong dependencies" if there is any constraint of resources in the project (people, time, money) in case, for example, during a maintenance activity. In addition, the major initiatives, according to Wang and others (2009) - LSI and VSM, ProbIR - are focused on NLP, which implies in a large number of false positives (Sundaram *et al.*, 2010). To minimize this problem, the approaches RTM-Fuzzy and RTM-N combine the RTM-NLP with the RTM -E presented in this paper, minimizing the number of false positives and increasing the effectiveness of the approaches, as shown by the experimental study.

The four approaches were implemented in the COCAR environment, so that experimental study could be performed to evaluate the effectiveness of each approach. The results showed that RTM-Fuzzy and RTM-N presented superior effectiveness compared to the others. The disadvantage of the approaches is that they are restrict to FRs.

The results motivate the continuity of this research, as well as further investigation into how better to combine the approaches for RTM creation using fuzzy logic. The main contributions of this particular work are the incorporation of the COCAR environment, which corresponds to the automatic relationship determination between FRs. This facilitates the evaluation of the impact that a change in a requirement can generate on the others. New studies are being conducted to improve the effectiveness of the approaches. As future work, it is intended to improve the NLP techniques. Another investigation to be undertaken relates to how an RTM can aid the software maintenance process, , offering support for regression test generation.

REFERENCES

Artero, A. O. 2009. Artificial intelligence - theory and practice. , 1st ed., São Paulo: Livraria Fisica.

Baeza-Yates, R., Berthier, A., Ribeiro-Neto, A. 1999. Modern Information Retrieval. 1st ed. New York: ACM Press / Addison-Wesley.

Cleland-Huang, J., Gotel, O., Zisman, A. 2012. Software and Systems Traceability. 1st ed., London: Springer.

Coppin, B. 2004. Artificial Intelligence Illuminated, 1st ed. Burlington: Jones and Bartlett Publishers.

Cuddeback, D., Dekhtyar, A., Hayes, J.H. (2010). Automated requirements traceability: the study of human analysts. In *18th IEEE International Requirements Engineering Conference*, RE. Sydney, Australia, Sep.: IEE Computer Society. 231-241.

Cysneiros, Zisman, A. (2008). Traceability and completeness checking for agent oriented systems. In *ACM Symposium on Applied Computing*, Fortaleza, Brasil, New York: ACM Digital Library. 71-77.

Deeptimahanti, D. K., Sanyal, R. (2011). Semi-automatic generation of UML models from natural language requirements. In *India Software Engineering Conf.*. Kerala, India, Feb. New York: ACM Digital Library. 165-174.

Deerwester, S., Dumais, S.T., Furnas, G. W., Landauer, T.K., Harshman, R. 1990. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6), 391–407.

Di Thommazo, A., Martins, M. D. C., Fabbri, S.C.P.F. (2007). Requirements management in COCAR environment (in portuguese). In *Requirements Engineering Workshop*, WER, Toronto, Canada, May. Rio de Janeiro: PUC-Rio. 11-23.

Di Thommazo, A., Malimpensa, G., Olivatto, G., Ribeiro T., Fabbri, S. (2012). Requirements traceability matrix: automatic generation and visualization. In *26th Brazilian Symposium on Software Engineering*, Natal, Brazil. May.: IEE Computer Society. 101-110.

Di Thommazo, A., Ribeiro, T., Olivatto, G., Rovina, R., Werneck, V., Fabbri, S. (2013). Detecting traceability links through neural networks. In *25th International Conference on Software Engineering and Knowledge Engineering*, SEKE, Boston, USA. July. Illinois: Knowledge Systems Institute, 2013. 36-41.

Goknil, A., 2011. Semantics of trace relations in requirements models for consistency checking and inferencing. Software and Systems Modeling, 31-54.

Guo, Y, Yang, M., Wang, J., Yang, P., Li, F. (2009). An ontology based improved software requirement traceability matrix. In *2nd International Symposium on Knowledge Acquisition and Modeling*, KAM, China, Dec. Los Alamitos: IIEE Computer Society. 160-163.

Hayes, J. H., Dekhtyar, A., Osborne, J. (2003). Improving requirements tracing via information retrieval. In *11th International IEEE Requirements Engineering Conference*, Monterey, CA, Sep. Los Alamitos: IEE Computer Society. 138-147.

Hayes, J. H., Dekhtyar, A., Sundaram, S. 2006. Advancing candidate link generation for requirements tracing: the study of methods. IEEE Transactions on Software Engineering, 32(1), 4–19.

Herrera F. 2008. Genetic fuzzy systems: taxonomy, current research trends and prospects. Evolutionary Intelligence , Volume 1, Issue 1 , pp 27-46 DOI 10.1007/s12065-007-0001-5.

IBM, Ten Steps to Better Requirements Management. [Online]. Available at: http://public.dhe.ibm.com/ common/ssi/ecm/en/raw14059usen/RAW14059USEN .PDF. [20 May 2012].

Kannenberg, A., Saiedian, A. 2009. Why software requirements traceability remains a challenge: CrossTalk. The Journal of Defense Software Engineering, 14-19.

Kawai, K. K., 2005. Guidelines for preparation of requirements document with emphasis on the functional requirements (in Portuguese). Master degree thesis. São Carlos, Brazil: Universidade Federal de São Carlos.

McCulloch , W. S., Pitts, W. 1943. A logical calculus of the ideas imminent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4), 115-133.

Real , R., Vargas, J. M. 1996. The probabilistic basis of Jaccard's index of similarity. Systematic Biology, [Online]. 45(3), 380-385. Available at: http://sysbio.oxfordjournals.org/content/45/3/380.full [Accessed 31 July 2013].

Salem, A. M., (2006). Improving software quality through requirements traceability models. In *4th ACS/IEEE International Conference on Computer Systems and Applications*, AICCSA, .Dubai, Sharjah, March, Los Alamitos: IEE Computer Society. 1159-1162.

Salton, G., Allan, J. (1994). Text retrieval using the vector processing model. In *3rd Symposium on Document Analysis and Information Retrieval*, Univ.of Nevada,

Sommerville, I. 2010. Software Engineering. 9th ed. New York: Addison Wesley.

Standish Group, CHAOS Report 2005, 2005. [Online]. Available at: http://www.standishgroup.com/sample _research/PDFpages/q3-spotlight. [Accessed 20 July 2013].

Standish Group, CHAOS Report 1994, 1994. [Online]. Available at: http://www.standishgroup.com/sample _research/chaos_1994_2.php. [Accessed 20 July 2013].

Sundaram, S. K. A. Hayes, J. H. B., Dekhtyar, A. C., Holbrook, E. A. D., (2010). Assessing traceability of software engineering artifacts. In *18th International IEEE Requirements Engineering Conference*, Sydney, Australia, Sep.: IEE Computer Society. 313-335.

Zadeh, L. A. 1965. Fuzzy sets. Information Control, 8(1),

338–353.
Zisman, A., Spanoudakis, G. 2004. Software Traceability: Past, Present & Future, Requirenautics Quarterly, 13, Newsletter of the Requirements Engineering Specialist Group of the British Computer Society, Sep.
Wang, X., Lai, G., Liu, C. 2009. Recovering relationships between documentation and source code based on the characteristics of software engineering. Electronic Notes in Theoretical Computer Science, 243(1),