Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Economics and Finance 00 (2012) 000–000

Procedia
Economics and Finance

www.elsevier.com/locate/procedia

NASBASE 2015

Error leakage and wasted time: Sensitivity analysis of a requirements consistency checking process

Jane Huffman Hayes[1*], Yann-Gaël Guéhéneuc[2], Wenbin Li[1], Miroslaw Truszczynski[1], Giulio Antoniol[2]

[1]*Computer Science Department, University of Kentucky, Lexington, Kentucky, 40506, USA*
[2] *Departement d'informatique, Ecole Polytechnique de Montreal, CP 6079, succ. Centre-Ville, Montreal, Quebec, H3C 3A7*

Abstract

A myriad of techniques are used by requirements engineering researchers and practitioners to address difficult problems, such as consistency checking of temporal requirements. Often, complex problems are addressed by building processes/tools that combine multiple techniques where the output from one technique becomes the input to the next technique, e.g., feature location that uses information retrieval and dynamic analysis techniques in sequence to perform the three step process of preparing a corpus, generating queries, and retrieving results. While powerful, these techniques are not without flaw. Inherent errors in each technique may leak into the subsequent step of the process. Errors then can be viewed as variations in the overall process. Errors of omission, or failure to retrieve elements, are viewed as error leakage because the "lost" elements will not be processed in subsequent steps. Errors of commission, or retrieval of irrelevant elements, amount to wasted time as human analysts will review/analyze these extraneous elements. As software quality professionals, developers, and researchers depend on these processes to verify and validate software and attendant artifacts, it is important to understand the impact of these errors on the quality of the output of the final step of the processes, e.g., the accuracy of the list of features retrieved using feature location. Therefore, we model and study one such process, for checking the consistency of temporal requirements. We study the process and assess error leakage and wasted time considering this process as fully automated. We perform an exploratory sensitivity analysis using Monte Carlo simulations of the input factors of our model to determine the effect that these sources of uncertainty, i.e., the errors of omission and commission, may have on the final accuracy of the consistency checking process. The sensitivity analysis uses published data on accuracy of previous techniques and data collected by applying the process on a real-world system. We share insights gained and discuss its applicability to other processes built of piped techniques.

Keywords: Sensitivity analysis; error propagation; requirements engineering; consistency checking; information retrieval; natural langauge processing; classification; semantic role labeling; error leakage; process model

1. Introduction

Challenges loom large as we endeavour to develop software systems. Many of these challenges are related to requirements engineering (RE) and its subareas of process, elicitation, specification, analysis, and validation [1]. In eliciting requirements, we

* Corresponding author. Tel.:+1-859-257-3171; fax: +1-859-323-3740.
E-mail address: hayes@cs.uky.edu.

face several obstacles as we attempt to ensure that all stakeholders are represented. In striving for thoroughness and exactness, we contend with customers and end users who do not understand formal languages or specifications. We also struggle to specify requirements that are complete and consistent and we deal with the inherent ambiguity of natural language text. As researchers, we strive to understand these and other issues and to develop useful approaches.

Requirements challenges can be addressed by various techniques, such as information retrieval, natural language processing, classification, clustering, to name a few. Complex requirements problems may be addressed by a collection of such techniques. For example, classifying a requirement as functional or non-functional requires this sequence of steps: perform pre-processing on the text of all of the requirement elements (remove stop words, spell out acronyms, etc.); tag each token in each requirement element with its part of speech; develop a training set to be used with a given dataset; and, use a classification or clustering technique to determine the labels for each requirement element (function or non-functional requirement).

The aforementioned techniques (and others) are powerful, but not perfect. When researchers or practitioners build complex processes or tools from these non-perfect techniques, errors may leak into subsequent process steps (analogous to defects leaking into subsequent software development phases). For example, it is well known that the typical information retrieval techniques used in tracing requirements achieve high recall but only low to moderate precision. Errors due to the imperfect and imbalanced precision and recall may seem insignificant, particularly if a requirements problem approach uses only one technique, such as requirement test coverage. But more complex requirements engineering problems require multiple techniques and, as in the functional/non-functional classification example given above, it is often the case that the output of one technique is the input to another technique, e.g., the output of parts of speech tagging becomes the input to the classification technique. Errors that leak from one technique to the next will impact the accuracy of the final result and will result in wasted time due to false positives introduced into the process.

Errors in the techniques have been the subject of much work but researchers focused on the quality of *individual* techniques, such as improving the recall and precision of retrieval techniques used for feature location. To the best of our knowledge, they did not study sequences of these techniques and error leakage. In this paper, we only consider fully-automated processes, in which human analysts do not intervene; future work includes modelling the impact of the human analyst-in-the-loop [2].

Researchers and practitioners often have a number of techniques from which to choose at the various steps of a requirements engineering process. For example, in the functional/non-functional requirement categorization example, they can select one among a multitude of clustering techniques. It seems intuitive that they should select the technique with the highest quality or accuracy, i.e., the lowest error rate or error leakage rate. However, accuracy has different definitions and values depending on the technique. Also, accuracy is influenced by the characteristics of the artifacts and/or the domain of the artifacts and, consequently, accuracy has a level of uncertainty. For example, the vector space model (VSM) with term frequency-inverse document frequency weighting (tf-idf) is used often for requirements tracing. In a comparison of studies on requirement tracing, we observed that VSM (not enhanced by thesauri or similar) achieved recall of 83% with precision of 54% [3]. In a different setting, using different datasets written using different vernacular, recall was merely 25.4% with precision of only 11.4% [4, 5]. This uncertainty must be taken into account.

With these error leaks and uncertainties, how is a researcher or practitioner to decide what techniques to use for the various steps of a requirements engineering process? How can a researcher know the factors that most influence the final quality of a requirements engineering process? We argue that search-based software engineering can support this decision process by applying sensitivity analysis to help guide the selection of techniques to minimize error leakage and wasted time. This paper is a first step toward that goal. Using the consistency checking of temporal requirements with Temporal Action Language (TeAL) [6] as our running example, we formulate the problem as a fully automated process—a piped sequence of techniques with the output of each serving as the input to the next. We perform exploratory sensitivity analysis, using published data on accuracy of techniques, simulated data, and data collected by applying TeAL on a real world system. We found that retrieval rates and precision values of the techniques influence the quality of the overall process.

This paper can be viewed as the discovery step prior to formulating our problem as a high dimensional problem requiring search. Our case is currently low-dimensional (the dimension is determined by the number of steps in the process). However, optimization of our eventual objective function, which will explicitly involve probability distributions as it is trying expected behaviours of the techniques on diverse datasets, will require search.

The paper is organized as follows. We present related work (Section 2) before modeling error leakage and wasted time (Section 3). We collect data about the accuracy of techniques through a literature review (Section 4). We discuss sensitivity analysis next (Section 5). Then, we present results and analysis (Section 6), followed by discussions and threats to validity (Section 7). Finally, we conclude with future work (Section 8).

2. Related Work

This section presents prior related work and shows that, although researchers studied various processes in software engineering, in particular the process of requirement consistency checking, and modeled errors for hardware systems, they did not study error leakage and its impact on the accuracy of final results and resulting wasted effort. Typically, previous work that studies software engineering processes, such as work by Le et al. [7], did not examine a sequence of techniques that together form a tool/process. Le et al. used a training set of data and built an SVM model, using part of the data for training and part for testing. We propose a model of the consistency checking process and then perform exploratory sensitivity analysis. The previous authors have pre-defined fault types for which they are predicting, thus their work represents a classifier [7]. Baker et al. [8] studied the selection of components to be released in a next version as a search-based problem. This work inspired us to use search-based methods to explore techniques to use in a process depending on their features, in terms of precision and recall.

2.1. Process Modeling

Alegria et al. describe a tool for building software process models, Avispa. Avispa is a graphical tool for analyzing Software Process Engineering Metamodel (SPEM) 2.0 [1] process models. The authors discuss the notion of error patterns that are commonly encountered in process modelling, such as Isolated roles (node not connected with an edge), Multiple Purpose tasks (too many output work products), and Overloaded Roles (role involved in too many tasks) [9]. However, they do not model or simulate the impact of these error patterns on the final results of the process.

Anan et al. examine the number of errors by phase of the software development lifecycle, using various models to estimate errors [10]. For example, they examine software reliability models, such as Goel-Okumoto, to examine failures during testing to estimate remaining undetected errors. They did not discuss the impact of the different techniques used during each phase on the overall errors and final results of the software development process.

Other research work modeling software engineering processes includes that of Pandey, Suman, and Ramani [11], of Krusche, Alperowitz, Bruegge, and Wagner [12], and of Sadiq, Mohd, and Hassan [13]. Although this previous work models various processes and provides evaluation of these processes, these researchers did not study the impact of errors at each step of the processes and the propagation of these errors through the whole process and their impact on final results.

2.2. Hardware Modeling

Ni et al. examine a new approach to predicting process variation’s impact on power leakage for a micro-pipeline circuit [14]. They identified many parameters as potential sources of process variation and represented them as random variables, assuming a Gaussian distribution. They varied the process parameters by + or – 10% of their mean. They examined the effect these parameters had on what they called the compact model parameter, which was representing the threshold voltage. They drew Pareto plots to understand the relative magnitude of effects the parameters have on the compact model parameter and used coefficient of determination to measure the goodness of the fit (R2 and R2 adjusted). They then built their process model using a CAD tool, compared its results (this was the predicted probability density function for leakage power) to histograms generated by 5000-sampled Monte Carlo simulation, and found a good fit.

We drew inspiration from this work to model and study requirement engineering processes as collections of piped techniques. To study our model of the processes, we simulate the impact of errors introduced by each technique and their leaking to the next technique to assess the overall impact of error leakage on the final results of the processes.

2.3. Consistency Checking

As the subject of this study, we chose the problem of requirement consistency checking because it is a well-known and difficult problem in requirements engineering, and because proposed solutions are processes made of several piped techniques. Typically, this problem arises because inconsistencies among software requirements must be removed to ensure the quality of software projects. Consistency checking tasks are time-consuming and error-prone if performed manually. Therefore, researchers proposed alternatives to manual checking using a formal system, such as Linear Temporal Logic [15], Answer Set Programming [16, 17], and Temporal Action Logic [18]. Once the researchers provided such a formal representation of requirements, they could use automated reasoning tools for the automated detection of inconsistencies.

Recently, a promising semi-automated approach was proposed to address that particular aspect of the process in the case of temporal requirements. The approach is based on the Temporal Action Language TeAL [6], developed by several of the co-authors of this paper, and on tools that process natural language requirements. A tracing tool, RETRO [19], is used to extract temporal requirements and some additional non-temporal related requirements from the requirement document. The extracted requirements are parsed and key elements of the system they specify are identified using the Stanford Parser [20, 21, 22]. Based on this information, an "almost TeAL" approximation of the sought TeAL representation is produced by a tool GenerateTeAL. What remains is for the analyst to correct (if necessary), complete, and verify the output of GenerateTeAL; this task has been demonstrated to be significantly easier for analysts than that of generating the TeAL theory directly from the requirement text.

In the following, consistency checking of requirements using the Temporal Action Language TeAL is the subject of our simulation and sensitivity analyses to show that errors leak from one technique to the next and that they impact the final results of the requirement consistency checking process.

3. Modeling Error Leakage and Wasted Time

We now describe our overall approach and present the formulation of the problem as an automated sequence of techniques for consistency checking of temporal requirements.

3.1. Towards Selecting the Optimal Techniques for a Consistency Checking Process

We use as the running example a consistency checking process that is composed of multiple sub processes, each implemented by one technique (we limit our scope to single techniques). Based on our industry experience and work with industrial partners, the results of such an automated process are not accepted at face value. Rather, some level of checking or review is performed by a human analyst before the results are used. It is intuitive that an automated process that retrieves all relevant elements and only relevant elements will result in the easiest review task for the human analyst. It is also intuitive that an automated process that fails to retrieve relevant elements and only retrieves irrelevant elements will require too much work and result in frustration for the analyst. However, these are black and white scenarios; studies show that techniques are not perfect [23] and that reality lies somewhere between these two extremes. How do we know what aspects of these techniques most influence the quality of the overall process?

To help researchers and practitioners select the “best” techniques for each sub process, we seek to minimize the amount of work required after the overall process has been performed. Toward that end, in this exploratory study, we seek to understand what most influences the quality of the overall consistency checking process. We follow a four-step approach. 1) We develop a model of the overall process that computes a set of output factors from a set of input factors (see Section 3.2). We combine the output factors into one objective function for the overall consistency checking process. 2) We determine the distributions for the input factors and use Monte Carlo simulation to randomly generate data according to the distributions, as shown in Figure 1. We use three of the four sources suggested by Wagner [24]: scientific literature, expert opinion where needed, and controlled experimental study of our Temporal Action Language (TeAL) process [6] (see Section 4). 3) We generate scatterplots of the input factors to determine their influence (see Section 6). Finally, 4) we use correlation analysis to perform an exploratory sensitivity analysis and to rank the model inputs using their contributions to output variability (also in Section 6).

3.2. Automated Process Problem Formulation

For our discussion, we assume that we work with the set $R = \{r_1, ..., r_n\}$ of n requirements specifying a software system. We denote by T the subset of R consisting of all temporal requirements in R and assume that $T = \{t_1, ..., t_m\}$.

We assume that to check the consistency of the temporal requirements in R, we use a process P comprising sub-processes to be performed in a sequential order: the input to P is sent as the input to the first process, its output is sent as input to the next process, etc. We write $P = < P_1, \ldots, P_k>$ to indicate the sub-processes forming P, as well as the order in which they are performed. We further assume that each step $\mathrm{P_i}$, $i = 1, ..., k$, can be accomplished by any of the several available j_i techniques in the set $C_i = <p_1, ..., p_j>$. We denote as p_k the technique chosen to fulfill the sub-process k.

Each technique has a retrieval rate ar. This is the recall of the technique, where recall is defined as the percentage of relevant elements that are retrieved over all retrieved elements. There is also an error rate er associated with the technique; this represents non-retrieved or “lost” elements. These elements can also be thought of as false negatives or Type 2 errors. Together they satisfy the identity *er* + *ar* =1. Thus, if an information retrieval technique, such as LDA, has recall of 0.9, its error rate is 0.1. If a noun-verb classifier has retrieval rate of 0.8, its error rate is 0.2. With *ar* and *er*, we limit our interest to lost elements, which likely will no longer be considered in the subsequent sub-processes due to the imperfect recall of upstream techniques. These lost elements due to errors leak through a process. We denote error leakage as *el*; for a given sub process *el* = *number of relevant elements* × *er*.

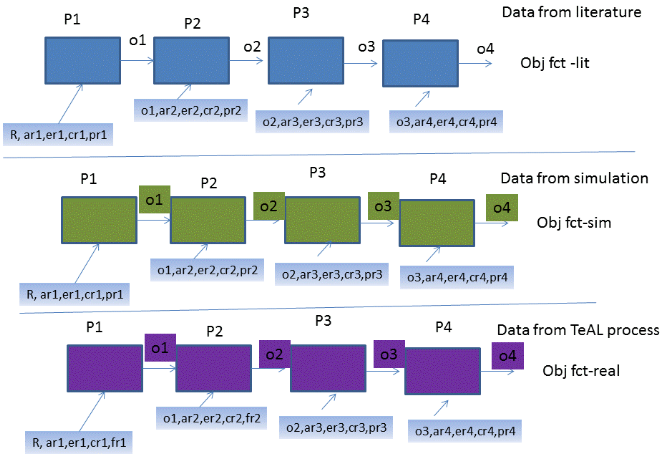

Fig. 1. Consistency checking process with error leakage/wasted time measures from three sources

Each technique has a "wasted time" cost rate *cr* due to irrelevant elements retrieved that must be processed in this and subsequent sub-processes. These elements can also be thought of as false positives or Type 1 errors. For techniques that are retrieving elements from a collection, the precision of the technique, or *pr*, is defined as the percentage of retrieved elements that are relevant over the retrieved elements. For classification techniques, *pr* measures how well elements are labelled or categorized. In either case, *pr* can be used to find *cr*. Together they satisfy the identity *cr* + *pr* = 1. Thus, if an information retrieval technique has precision of 0.6, *cr* is 0.4. We are interested in the false positive elements as these connote possible wasted time/extra effort in subsequent sub-processes. We denote wasted time as *wt*; for a given sub process, *wt* = *number of relevant elements* × *cr*.

The objective function (*process-cost*) that we will use in performing the sensitivity analysis combines these error leakages and wasted times. We quantify the extra work for an analyst examining the output of a given process as the time that must be invested to find lost elements (*el*) plus time that must be invested to examine elements that should not have been retrieved (*wt*):

$$\textit{process-cost} = \mathit{el}(\mathrm{p}_k) + \mathit{wt}(\mathrm{p}_k).$$

Running Example: In our running example, we examine the four sub-processes of consistency checking of temporal requirements, i.e., $i = 4$. Let us assume a document *R* describing 500 requirements, $R = \{r_1, \ldots, r_{500}\}$, 100 of which, denoted t_1, …, t_{100}, are temporal, i.e., $T = \{t_1, \ldots, t_{100}\}$. We apply a four-step consistency checking process *P* to analyze *R*, $P = < P_1, P_2, P_3, P_4>$. Let us assume that each step P_i, i = 1, 2, 3, 4, can be accomplished by a technique p_i with accuracy rate ar_i, cost cr_i, and precision pr_i. We model and study the quality of the output of *P* 4, that is, the output of the entire process *P*. This quality is expressed by our objective function, which depends on the parameters of the techniques selected for each step.

Sub-process P_1 determines the set *T* consisting of temporal requirements in *R*. It is accomplished by the technique p_1, which maps any set of requirements *R* to its subset $o_1 = p_1(R)$. Ideally, o_1 will consist of all temporal requirements in *R*. However, in performing this mapping, some temporal requirements may not be retrieved from *R* and we quantify this error with the error rate er_1, and some non-temporal requirements will be retrieved, which we quantified by the precision pr_1. In our notation and recalling that $ar = 1 - er$ and $cr = 1 - pr$, we have:

$el(p_1) = n \times er_1$
$wt(p_1) = n \times cr_1$.

The goal of the second sub-process P_2 is to identify in R non-temporal requirements that are related to the temporal ones and so might affect the consistency of the temporal requirements. That sub-process takes as input the original set R of requirements less the set o_1 of requirements identified as temporal in Step 1. The set o_1 is only an approximation of T. In this example, the sub process P 2 is accomplished by the technique p_2 and outputs the set o_2.

In performing this transformation, some related requirements may not be retrieved due to the error er_2 of the technique p_2, and some unrelated ones may be depending on the precision pr_2 of the technique. Taking these rates into account and applying them to o_1 and R, we have:

$el(p_2) = (cardinality(R) - cardinality(o_1)) \times er_2$
$wt(p_2) = (cardinality(R) - cardinality(o_1)) \times cr_2.$

Sub-process P 3, accomplished by technique p_3, is applied to each of the requirements in the sets o_1 and o_2 to tag their parts of speech. The output consists of a set o_3 of tags. In performing this transformation, we use only the retrieval rate ar_3 for this sub-process and set pr_3 to 0 because one and only one tag is assigned per part of speech by definition of the requirement consistency checking problem. We thus assume that all tags in a requirement are correct or all are incorrect. So, given a collection of 10 requirements that are tagged with 0.8 ar_3, we assume that 8 requirements are correctly tagged and 2 are not. We thus have:

$el(p_3) = (cardinality(o_1) + (cardinality(R) - cardinality(o_1))) \times er_3$
$wt(p_3) = wt(p_2) + pr_3.$

Finally, sub-process $P4$, achieved by technique p_4, is applied to the tagged temporal and related requirements retrieved from p_3, o_3, to identify the semantic roles in each requirement (such as action, agent). $P4$ transforms the output of $P3$ to a set of roles for each requirement tagged in $P3$. In performing this transformation, some roles may not be retrieved due to er_4, some may be retrieved that should not be, because of pr_4. We thus have:

$el(p_4) = cardinality(o_3) \times er_4$
$wt(p_4) = cardinality(o_3) \times cr_4.$

Finally, the overall objective function is:

$process\text{-}cost = el(p4) + wt(p4).$

where higher values of *process-cost* indicate more work for the analysts, lower *process-cost* values are desired.

4. Technique Accuracy

Information retrieval (IR) techniques have been used in a variety of process/tools supporting requirements engineering and software engineering, most notably in requirements tracing. To model the distribution of the input variables for our model, i.e., the set of error and cost rates of the used techniques, we sought published quality levels for existing techniques. We undertook a limited literature review, using appropriate search terms in the Google search engine as well as ACM Digital Library to retrieve articles, URLs, and related sources of information on IR technique accuracy. In Table 1, we show the values, summarized as lower and upper bounds for some techniques, as well as the source of the data and the dataset under evaluation when the measures were captured. For example, VSM with tf-idf weighting has a *pr* value of 0.276 with *ar* of 0.829 for the LEDA dataset. VSM has a lower bound for *er* of 0.000 and an upper bound of 0.746, while *ar* ranged from 0.254 to 1.000. The set of IR techniques have a lower bound of 0.110 and an upper bound of 0.540 for *pr*. Table 1 focuses on IR techniques because they are used to accomplish sub-processes P1 and P2 in our running example of consistency checking.

Table 1. Measures for IR techniques

	er	*ar*	*cr*	*pr*	Lit. Source	Dataset
VSM tf-idf	0.746	0.254	0.886	0.114	[4]	MODIS Albergate
	0.000	1.000	0.862	0.138	[25]	Albergate
	0.500	0.500	0.567	0.433	[25]	LEDA
	0.276	0.724	0.829	0.171	[26]	eTour
	0.100	0.900	0.830	0.170	[27]	
[low, upp]	[0.000, 0.746]	[0.254, 1.000]	[0.567, 0.886]	[0.114, 0.433]		
LDA	0.400	0.600	0.890	0.110	[27]	eTour
LSI	0.000	1.000	0.836	0.164	[25]	Albergate
	0.140	0.860	0.788	0.212	[25]	Albergate
	0.000	1.000	0.882	0.118	[25]	LEDA
	0.167	0.833	0.460	0.540	[25]	LEDA
	0.100	0.900	0.830	0.170	[27]	eTour
[low, upp]	[0.000, 0.167]	[0.833, 1.000]	[0.460, 0.882]	[0.118, 0.540]		
Jensen-Shannon	0.090	0.910	0.830	0.170	[27]	eTour
IR summary values [low, upp]	[0.000, 0.746]	[0.254, 1.000]	[0.110, 0.540]	[0.110, 0.540]		

Parts of speech (POS) taggers have been used in a variety of process/tools supporting requirements engineering and software engineering, most notably in requirements analysis, in particular for consistency checking. We examined the literature and found the following information on the accuracy of a number of POS tagging techniques. In Table 2, we show the lower and upper bounds for the measures as well as the source of the data and the dataset from which the values were obtained. For example, the CLAWS parser has a lower bound of 0.500 for *er* and an upper bound of 0.400; a lower bound of 0.950 for *ar* and an upper bound of 0.960 when applied on the Wall Street Journal (WSJ) subset of the Penn Treebank dataset. POS tagging techniques are used to accomplish sub-process P3 during consistency checking.

Table 2. Measures for POS tagging techniques

	ar	Lit. Source	Dataset
OpenNLP	0.966	[28]	OpenNLP training data
Stanford Parser	0.973	[29]	Penn Treebank WSJ
CLAWS	[0.950, 0.960]	[30, 31, 32, 33, 34]	Penn Treebank WSJ
LBJ	0.966	[35]	Wall Street Journal corpus
Minipar	0.800	[36]	SUSANNE corpus
POS summary values [low, upp]	[0.800, 0.973]		

Semantic role labellers are used more and more frequently as researchers and practitioners alike realize that syntactic analysis of natural language text will not suffice for today's applications. These role labellers glean semantic information from natural language text by ascribing meaning, in a contextual setting, to tokens in texts. These techniques support processes/tools such as translation of requirements to LTL. We examined the literature and found the following information on the accuracy of a number of SRL techniques. In Table 3, we show the lower and upper bounds for the measure and the source of the data and the datasets used to collect the values. For example, Senna has a lower bound of 0.55 for *er* and an upper bound of 0.57; a lower bound of 0.43 for *ar* and an upper bound of 0.45 when applied on biomedical texts. SRL are used to accomplish sub-process P4 during consistency checking.

Table 3. Measures for SRL techniques

	er	*ar*	*cr*	*pr*	Lit. Source	Dataset
Senna	[0.55,0 0.570]	[0.430, 0.450]	[0.320, 0.290]	[0.680, 0.710]	[37]	Biomedical texts
ISRL	0.368	0.630	0.324	0.666	[42]	WSJ from PropBank
NLPNet SRL	0.302	0.598	0.328	0.662	[41]	PropBank-Br
SRL summary [low, upp]	[0.302, 0.570]	[0.430, 0.630]	[0.320, 0.328]	[0.662, 0.710]		

Finally, consistency checking of temporal requirements using the TeAL technique was studied and applied to the 511 phone system in California [38]. During a previous study [39], we collected the same measures described above, the values are shown in Table 4. Our approach uses the techniques introduced above: VSM, Stanford parser, and Senna. The sub-process P1, identifying temporal requirements, has 0.288 for *er*, 0.712 for *ar*, 0.516 for *er*, and 0.484 for *pr*. The sub-process P2, identifying non-temporal related requirements, has 0.559 for *er*, 0.441 for *ar*, 0.516 for *er*, and 0.484 for *pr*. The sub-process P3, tagging parts of speech, has 0.143 for *er* and 0.857 for *ar*. The sub-process P4, identifying semantic roles (predicates and arguments), has 0.130 for *er*, 0.870 for *ar*, 0.167 for *cr*, and 0.833 for *pr* when we use Senna to find predicates, and has 0.385 for *er*, 0.615 for *ar*, 0.294 for *cr*, and 0.706 for *pr* when we use Senna to find arguments.

Table 4. Measures for TeAL technique

	er	*ar*	*cr*	*pr*	Lit. Source	Dataset
P1 – VSM	0.288	0.712	0.516	0.484	Unpublished notes	Phone 511
P2 – VSM	0.559	0.441	0.516	0.484	Unpublished notes	Phone 511
P3 – Stanford	0.143	0.857			[39]†	Phone 511
P4 – Senna	[0.130]‡ [0.385]‡	[0.870]‡ [0.615]‡	[0.167]‡ [0.294]‡	[0.833]‡ [0.706]‡	Unpublished notes	Phone 511

5. Sensitivity Analysis

A sensitivity analysis is used to assess the impact of variations in the inputs of a mathematical model on its outputs to understand the impact of uncertainty in the inputs on the results of the model as well as assessing the robustness of the model, i.e., does the model produce aberrant results if the inputs vary slightly or greatly? Sensitivity analysis provides a ranking of the model inputs based on their relative contributions to model output variability and uncertainty [40]. In this study, we seek to assess the impact of the variation of input factors *ar* and *pr* on the overall output (which, due to the identity relationships, in turn tells us about *er* and *cr*), *process-cost*, of the requirement consistency checking process.

Our research approach is described in Section 3.1. Here we focus on research steps two, three, and four (step one, developing the model, was described in Section 3.2). In the second step, we determined the distribution of *ar* and *pr* for each sub-process. We generated data for the sub-processes *P*1 ar_1 and pr_1 values, *P*2 ar_2 and pr_2 values, *P*3 ar_3 value, and *P*4 ar_4 and pr_4 values in three cases: a) we took the data provided in Tables 1 through 4 in Section 4 and used the lower bound as a worst case, upper bound as a best case, and modelled a uniform distribution; b) we assumed a mean of 0.5 for all values and assumed a normal distribution; and c) we used the values from our empirical study on TeAL [6] and considered them as the means of a normally distributed distribution and used expert opinion to estimate variance (plus or minus 10%). We used the Monte Carlo method to generate 500 samples for each case, varying one factor at a time, keeping all other measures set at their mean values, e.g., the mean of the values from literature. This data serves as the input for step three of our research approach.

† We found Stanford parser to be 0.937 accurate.
‡ Predicates on the first line and arguments in the second line.

In step three, we generate XY scatter plots—for each input factor and the objective function with all the samples. We apply a virtual trend line to the scatter plots and analyze them. We look for possible unexpected plots that may indicate an error in the model. We also look for interesting trends that may indicate that one factor is highly correlated with the objective function, i.e., the output factors.

In step four of our research approach, we use correlation analysis to rank the input factors for the model and examine their relative contribution to our objective function. We use Pearson's Correlation Coefficient, *r*, a common measure of sensitivity, which measures the strength and direction of linear association between two quantitative variables.

6. Results and Analysis

Our first step was to identify the factors of our model, discussed in Section 3.2. Next, we generate distributions for each input factor. We then generate scatterplots for each of our data distribution sources and the output objective function, Process-cost. In Figure 2 below, we show the scatter plots for the literature values for all seven factors.

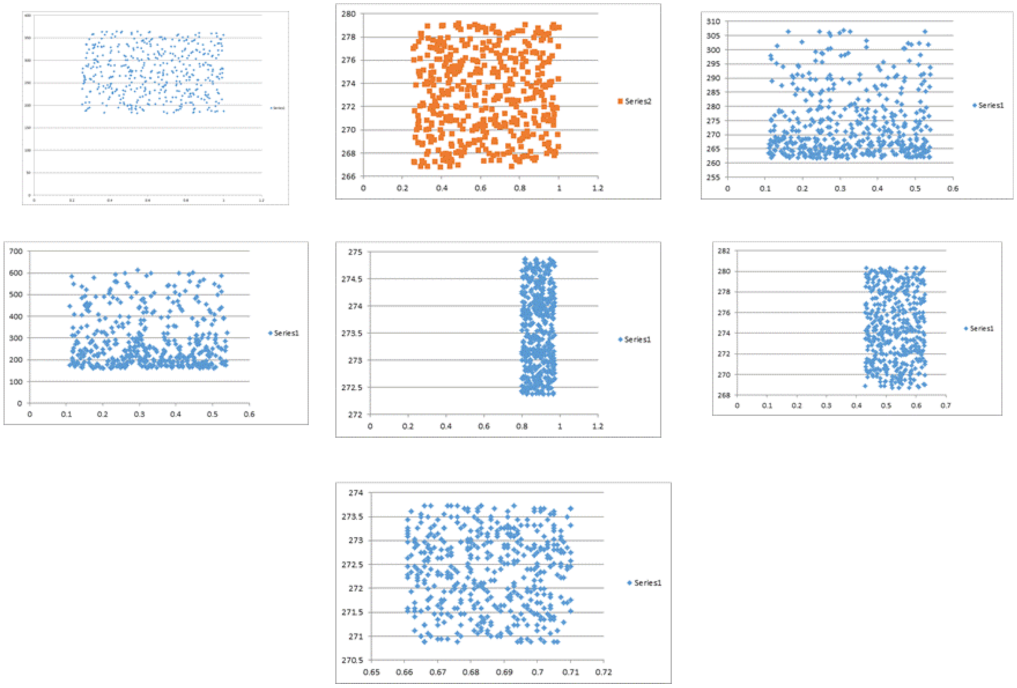

Fig. 2. X Y Scatterplot for literature source values for (from top left to bottom center): *ar* of sub-process *P*1, *pr* of sub-process *P*1, *ar* of sub-process *P*2, *pr* of sub-process *P*2, *ar* of sub-process *P*3, *ar* of sub-process *P*4, and *pr* of sub process *P*4

The data is quite chaotic for *P*2 *ar* and *pr* as well as *P*4 *pr*. There is a clear lower bound for *P*1 *ar* around 190 and an upper bound of 150. Similarly *P*2 *pr* has a lower bound of 260 with an upper bound of approximately 300. *P*3 *ar* has a very narrow columnar cluster around the value of 0.9, which yields an objective function in the narrow range of 272.4 and 274.75. Similarly, *P*4 *ar* exhibits a columnar shape between 0.450 and 0.625, which yields objective function values visually bounded by 268 and 280. When considering values from the literature the scatterplots show that the retrieval rates and precision values of the techniques are quite chaotic and yield, in the fourth step of the process, large variations and thus uncertainty on the output of the overall process. They thus confirm observations in the literature and our intuitions that all techniques are not equal per se and that they introduce variations in the techniques; also, but the *ar* and *pr* values depend on the datasets on which they are applied. We thus confirm that errors leak from one technique to the next, resulting in overall errors and wasted time for the analysts.

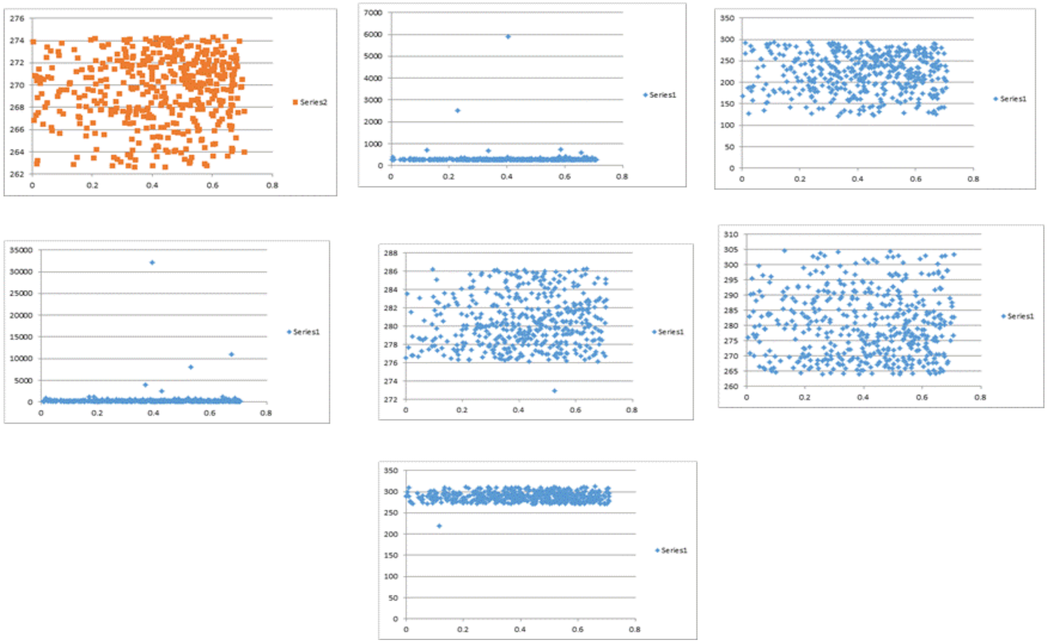

Fig. 3. X Y Scatterplot for normally distributed values around mean 0.5 for (from top left to bottom center): *ar* of sub-process *P*1, *pr* of sub-process *P*1, *ar* of sub-process *P*2, *pr* of sub-process *P*2, *ar* of sub-process *P*3, *ar* of sub-process *P*4, and *pr* of sub-process *P*4

Figure 3 shows the scatterplots for the synthetic data generated using a normal distribution with a mean of 0.5. *P*1 *ar* ranges between 262 and 274. *P*1 *pr* is tightly clustered at 272. *P*2 *ar* is bounded between 125 and 290. *P*2 *pr* is bounded tightly at 293. In contrast, *P*3 *ar* values range from 276 to 286. *P*4 *ar* appears much more chaotic, but is bounded by 265 (lower bound) and 305 (upper bound). Finally, *P*4 *pr* is tightly clustered at 288. Looking at the scatterplots, and in contrast to those shown in Figure 2, we observe that the *ar* and *pr* values of the last step of the process have fewer variations in precision *pr* but are still chaotic in *ar*. This confirms that variations introduced by one technique will leak into subsequent techniques and the final outcome of the process: subsequent techniques do not absorb much of the errors leaked from previous ones.

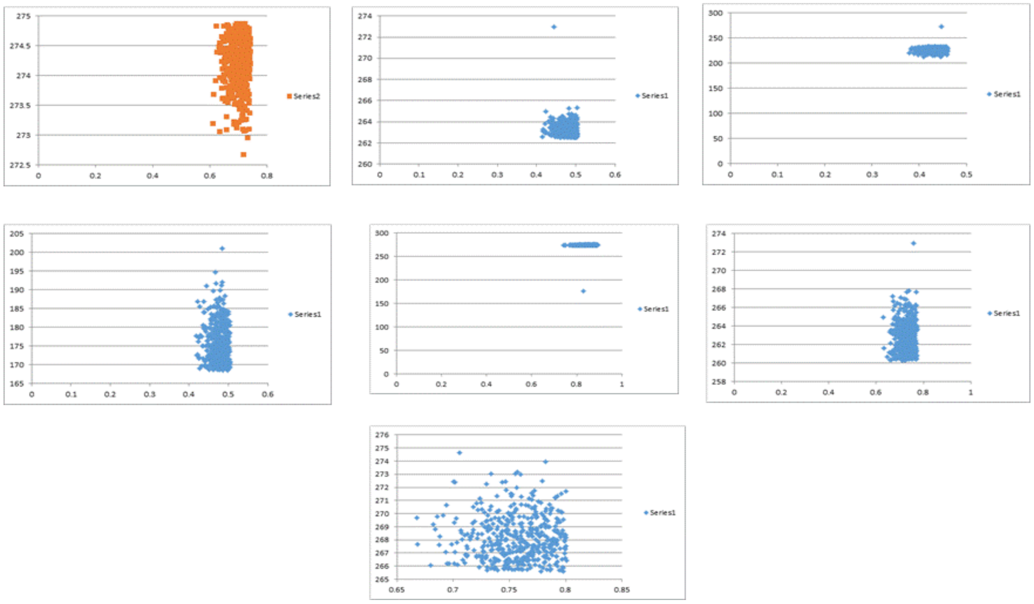

Fig. 4. X Y Scatterplot for TeAL study values for (from top left to bottom center): *ar* of sub-process *P*1, *pr* of sub-process *P*1, *ar* of sub-process *P*2, *pr* of sub-process *P*2, *ar* of sub-process *P*3, *ar* of sub-process *P*4, and *pr* of sub-process *P*4

Figure 4 shows the scatterplots for the data for the TeAL study. *P*1 *ar* ranges from 273 to 274.9. *P*1 *pr* is clustered tightly around 263 (and 0.470 for *pr*). *P*2 *ar* is tightly clustered at 226 and 0.433 for *pr*. *P*2 *pr* is columnar from 169 to 185. *P*3 *ar* is also tightly clustered, but at 274. *P*4 *ar* is columnar from 260 to 267. *P*4 *pr* is chaotic, though it appears to be bounded between 265.5 and 271. We observe that the values for *pr* in the last step are chaotic while those in pervious steps are much less so, thus hinting at a kind of amplification process in which few chaotic values from earlier sub-processes yield to chaotic values in the last sub-process. This observation does not occur for *ar*.

After generating scatterplots for all factors, we perform a correlation analysis and generate bar graphs showing the ranked correlation coefficients. These graphs (a.k.a. tornado plots or tornado charts) show both the relative magnitude and direction of influence (positive or negative) of each input factor (ar_i and pr_i with $i = 1, \dots, 4$) on the value of the objective function.

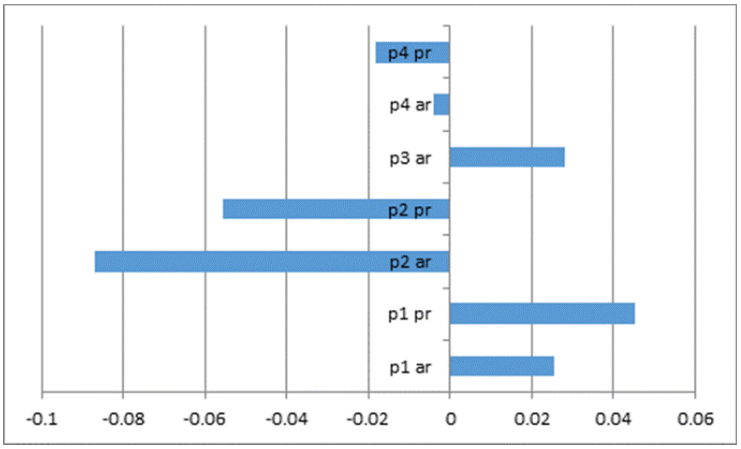

Fig. 5. Tornado chart for literature values of ar and pr for all subprocesses

Figure 5 shows that *P*2 *ar* has the largest influence by far, in the negative direction. *P*2 *pr* and *P*1 *pr* are close in magnitude, but in opposite direction. *P*1 and *P*3 *ar* are similar in magnitude and direction. We expected that *P*2 *ar* would be similar to *P*1 and *P*3 *ar*, but instead it has larger influence and in the opposite direction. None of the correlation coefficients are particularly large, the largest being -0.087.

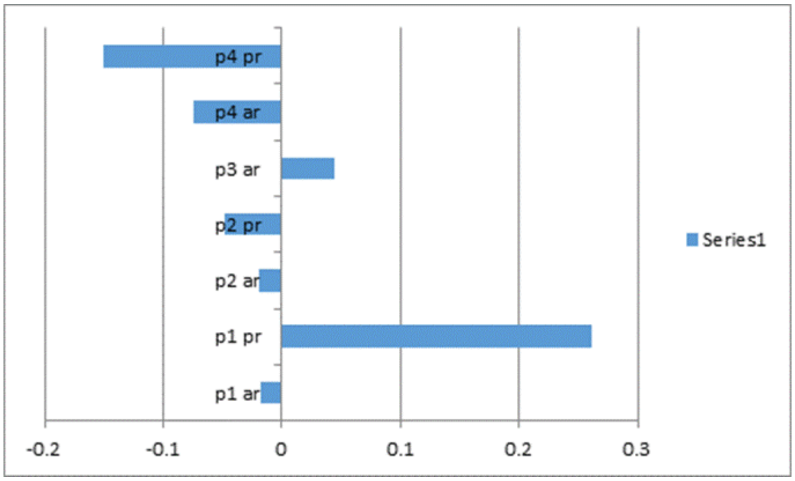

Fig. 6. Tornado chart for values around 0.5 mean for all subprocesses

Figure 6 shows that *P*1 *pr* has the most influence by far, in the positive direction, when considering values around 0.5 mean. *P*4 *pr* has the next most influence, in the negative direction. *P*2 *ar* and *P*3 *ar* are the next largest, but in opposite directions. *P*1 *ar* and *P*2 *ar* appear similar in magnitude and both in the negative direction. These coefficients are not much larger than for the literature data source.

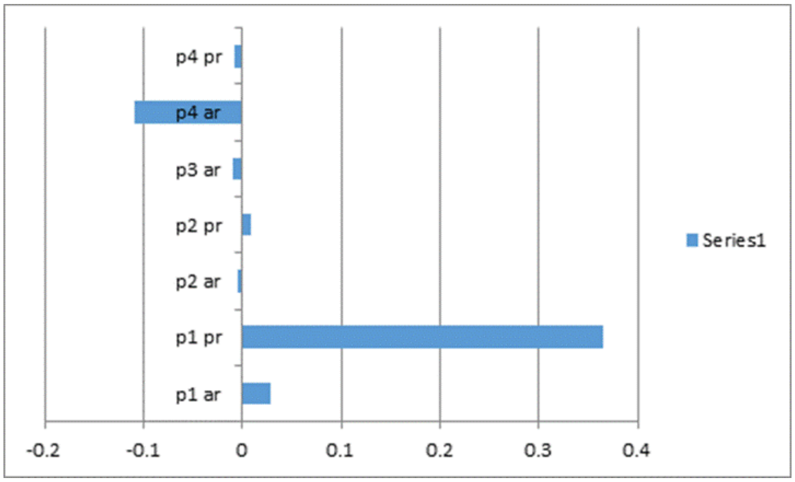

Fig. 7. Tornado chart for TeAL study values for all subprocesses

Finally, Figure 7 shows the tornado chart for the TeAL study values. Only *P*1 *pr* and *P*4 *ar* have any magnitude of mention. *P*1 *pr* is by far the largest influence, in the positive direction. *P*4 is much smaller, in the negative direction. The other coefficients are very small. Also, these coefficients are in line with the 0.5 mean data source and much larger than for the coefficients for the literature values.

7. Discussions and Threats to validity

Based on the exploratory sensitivity analysis, we can provide a relative impact of the input factors on the objective function as follows:

- for literature values: *P*2 *ar* > *P*2 *pr* > *P*1 *pr* > *P*3 *ar* > *P*1 *ar* > *P*4 *pr* > *P*4 *ar*
- for mean of 0.5: *P*1 *pr* > *P*4 *pr* > *P*4 *ar* > *P*2 *pr* > *P*3 *ar* > *P*2 *ar* > *P*1 *ar*
- for TeAL study: *P*1 *pr* > *P*4 *ar* > *P*1 *ar* > *P*3 *ar* > *P*2 *pr* > *P*4 *pr* > *P*2 *ar*

We thus observe that the influencing factors are different when considering the literature and when considering values centered on the 0.5 mean or values from the TeAL study. We further confirm that, in the literature, there are variations due to various techniques and the datasets on which they were used. We also observe that the last two data sources have similar factors in the first three positions: *P*1 *pr* and *P*4 *ar* both appear. *P*1 *pr* also appears in the first three slots for the literature value source. *P*4 *pr* appears in the next to last slot for two of the three sources. *P*1 *ar* is in the last half of the slots for two of the three sources. *P*2 *pr* and *P*3 *ar* appear in the fourth positions. Based on this, we conclude that, as expected, the *pr* and *ar* values of the sub processes all impact the outcome of the process (the value of the objective function) but with different effect. In particular, the precision of the first technique p_1, to determine the set *T* consisting of temporal requirements, is particularly important: it greatly impacts the results of the overall process. We expected such an observation because if this set contains too many false positives, then other techniques will more likely make mistakes and propagate useless requirements, increasing the human analysts' effort at the end.

The precision and retrieval rate of p_4 impact the overall results, possibly because p_4 is the last technique. At that point, any misclassified element cannot be reclassified correctly. In contrast, *P*2 and *P*3 impact the results less, except when considering the *pr* and *ar* values of associated techniques in the literature: the technique chosen has a great impact when it comes to finding related requirements and to tagging parts of speech.

We thus showed that retrieval rates and precision values of the techniques influence the quality of the overall process. These results call for further work now to decide what techniques to use for the various steps of a requirements engineering process. They should also be further analysed and refined to account for some threats to their validity.

Construct validity is threatened by our choice of considering only retrieval rates and precision values in our model. It is possible that a determinant factor in the results of a process is the dataset or some other unknown factor. We limited this threat by performing three Monte Carlo analyses using three sets of values: from the literature, centered around 0.5 mean, and from the TeAL study. Yet, future work should try to consider other factors.

Content validity is threatened by our use of lower and upper bounds from the literature. Getting values for many of the techniques from the literature (see Section 4) and using those literature values to "limit" our search threaten the content of our study because these bounds may artificially constraint the search. We limited this threat by considering many different works, from different authors and on different datasets. Yet our chosen references form a survey of convenience because we used as our search strings a set of keywords related to the validated accuracy of each technique category and did not perform a systematic literature review.

Internal validity of the obtained results is not threatened because our model accounts only for the *ar* and *pr* values of the different techniques used in the requirement consistency checking process.

Conclusion validity seems high because there is intuitively a relation between the results obtained from an overall process and the *ar* and *pr* values of the techniques used in this process.

External validity is threatened by our choice of the requirement consistency checking process as the running example. The results and conclusions obtained here may not apply to different requirements processes. Future work is necessary to replicate our study on other processes, made of other techniques, with other *ar* and *pr* values.

8. Conclusions and Future Work

Various techniques exist to build processes to solve problems in requirement engineering, such as information retrieval, natural language processing, classification, clustering. Complex requirements problems are typically addressed using a collection of such techniques piped together. Existing techniques are powerful, but not perfect, and errors leak from one technique to the next much as defects leak into subsequent software development phases. Errors due to the imperfect and imbalanced precision and recall of piped techniques will impact the accuracy of the final result and will result in wasted time due to false positives introduced into the process.

Errors in the techniques have been the subject of much work but researchers focused on the quality of *individual* techniques, such as improving the recall and precision of retrieval techniques. In this paper, we presented a first study of the error leakage between techniques and its impact on the accuracy and wasted effort resulting from this error leakage. We showed, through a Monte Carlo simulation and exploratory sensitivity analysis of a consistency checking process, that error leakage indeed impacts the accuracy and effort of a human analyst.

Following the study presented in this paper, we will move in several directions: one direction is to learn the features that are inherent in software artifacts and lead to errors in the techniques. For example, we can attempt to learn what inherent characteristics of a text lead it to have incorrect parts of speech tags assigned. By having many training examples, we can move toward developing a decision support assistant that will assist with configuration from the learned examples. We will take into consideration simulated or synthetic data. The next direction is to consider the case of requirements or elements where some of the POS tags or semantic roles are correct and some are not, rather than using simplifying assumptions. We plan to model the impact of the human analyst when some requirement engineering process has a human-in-the-loop. Finally, we plan to perform a systematic literature survey on techniques used in software engineering processes in general and requirement engineering processes in particular. This future work will help us in answering the more general question: How are we to know what techniques to select for each sub process?

Acknowledgments

We thank Drs. Stromberg and Bathke, and Joshua Lambert of the Statistics Dept. for reviewing the paper.

References

[1] OMG. 2008. Software Process Engineering Metamodel SPEM 2.0 OMG Specification. Technical Report ptc/07-11-01,OMG.

[2] Dekhtyar, A., Dekhtyar, O., Holden J., Hayes, J. H., Cuddeback, D., Kong, W. K., 2011. On Human Analyst Performance in Assisted Requirements Tracing: Statistical Analysis. In Requirements Engineering Conference (RE), 19th IEEE International, p. 111-120.

[3] Hayes, J. H., Dekhtyar, A., 2005. A Framework for Comparing Requirements Tracing Experiments. In International Journal of Software Engineering and Knowledge Engineering, 15(05), p.751-781.

[4] Hayes, J. H., Dekhtyar, A., Osborne, J. 2003. Improving Requirements Tracing via Information Retrieval. In Requirements Engineering Conference. Proceedings. 11th IEEE International, p.138-147. IEEE

[5] Hayes, J. H., Dekhtyar, A., Sundaram, S. K. 2006. Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods. In Software Engineering, IEEE Transactions on, 32(1), p. 4-19.

[6] Li, W., Hayes, J. H., Truszczyński, M. 2012. Temporal Action Language (TAL): A Controlled Language for Consistency Checking of Natural Language Temporal Requirements. In NASA Formal Methods, p. 162-167. Springer Berlin Heidelberg.

[7] Le, T. D. B., Lo, D., 2013. Will Fault Localization Work for These Failures? An Automated Approach to Predict Effectiveness of Fault Localization Tools. In Software Maintenance (ICSM), 29th IEEE International Conference, p.310-319.

[8] Baker, P., Harman, M., Steinhofel, K., Skaliotis, A., 2006. Search Based Approaches to Component Selection and Prioritization for the Next Release Problem. In Software Maintenance (ICSM), 22nd IEEE International Conference, p.176-185.

[9] Hurtado A., J. A., Bastarrica, M. C., Bergel, A. 2011. Analyzing Software Process Models with AVISPA. In Proceedings of the 2011 International Conference on Software and Systems Process, p. 23-32. ACM.

[10] Iwata, K., Nakashima, T.. Anan, Y.. Ishii, N.. 2007. Errors Estimation Models for Embedded Software Development Projects. In Software Engineering Research, Management & Applications, 2007. SERA 2007. 5th ACIS International Conference. p.815,822.

[11] Pandey, D., Suman, U., Ramani, A. K. 2010. An Effective Requirement Engineering Process Model for Software Development and Requirements Management. In Advances in Recent Technologies in Communication and Computing (ARTCom), International Conference, p. 287-291. IEEE

[12] Krusche, S., Alperowitz, L., Bruegge, B., Wagner, M. O. 2014. Rugby: an Agile Process Model Based on Continuous Delivery. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, p. 42-50. ACM.

[13] Sadiq, M., Hassan, T. 2014. An Extended Adaptive Software Development Process Model. In Issues and Challenges in Intelligent Computing Techniques (ICICT), International Conference p. 552-558. IEEE.

[14] Ni, C., Tarawneh, Z. A., Russell, G., Bystrov, A. 2012. Statistical Leakage Power Modelling of Manufacturing Process Variations at System Level. In Power Engineering and Automation Conference (PEAM), 2012 IEEE p. 1-4. IEEE.

[15] Huth, M., Ryan, M. 2004. Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press.

[16] Marek, V. W., Truszczyński, M. 1999. Stable Models and an Alternative Logic Programming Paradigm. In The Logic Programming Paradigm p. 375-398. Springer Berlin Heidelberg.

[17] Niemelä, I. 1999. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. In Annals of Mathematics and Artificial Intelligence, 25(3-4), p. 241-273.

[18] Doherty, P., Gustafsson, J., Karlsson, L., Kvarnström, J. 1998. TAL: Temporal Action Logics Language Specification and Tutorial. In Computer and Information Science, 3(015).

[19] Hayes, J. H., Dekhtyar, A., Sundaram, S. K., Holbrook, E. A., Vadlamudi, S., April, A. 2007. REquirements TRacing On target (RETRO): Improving Software Maintenance through Traceability Recovery. In Innovations in Systems and Software Engineering, 3(3), p. 193-202.

[20] Klein, D., Manning, C. D. 2003. Accurate Unlexicalized Parsing. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1 p. 423-430. Association for Computational Linguistics.

[21] De Marneffe, M. C., MacCartney, B., Manning, C. D. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In Proceedings of LREC Vol. 6, p. 449-454.

[22] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P. 2011. Natural Language Processing (Almost) from Scratch. In The Journal of Machine Learning Research, 12, p. 2493-2537.

[23] Kong, W. K., Hayes, J. H., Dekhtyar, A., Dekhtyar, O. 2012. Process Improvement for Traceability: A study of Human Fallibility. In Requirements Engineering Conference (RE), 20th IEEE International p. 31-40. IEEE.

[24] Wagner, S. 2007. Global Sensitivity Analysis of Predictor Models in Software Engineering. In Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops. International Workshop on p. 3-3. IEEE.

[25] Marcus, A., Maletic, J. I. 2003. Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing. In Software Engineering, Proceedings. 25th International Conference on p. 125-135. IEEE.

[26] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E. 2002. Recovering Traceability Links between Code and Documentation. In Software Engineering, IEEE Transactions on, 28(10), p. 970-983.

[27] Oliveto, R., Gethers, M., Poshyvanyk, D., De Lucia, A. 2010. On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery. In Program Comprehension (ICPC), 2010 IEEE 18th International Conference on p. 68-71. IEEE.

[28] Baldridge, J. 2005. The Opennlp Project. URL: http://opennlp. apache. org/index. Html

[29] Bajwa, I. S., Lee, M., Bordbar, B. 2012. Resolving Syntactic Ambiguities in Natural Language Specification of Constraints. In Computational Linguistics and Intelligent Text Processing p. 178-187. Springer Berlin Heidelberg.

[30] Chen, K. J., Huang, C. R., Chang, L. P., Hsu, H. L. 1996. Sinica Corpus: Design Methodology for Balanced Corpora. Language, p.167-176.

[31] Lua, K. T. 1996. Part of Speech Tagging of Chinese Sentences using Genetic Algorithm. In Proceedings of ICCC96, National University of Singapore, p. 45-49.

[32] Liu, S. H., Chen, K. J., Chang, L. P., Chin, Y. H., 1995. Automatic Part-of-Speech Tagging for Chinese Corpora. In Computer Proceeding of Oriental Languages, Hawaii, Vol. 9, p.31-48.

[33] Tsai, Y. F., Chen, K. J. 2003. Reliable and Cost-Effective PoS-Tagging. In CLCLP.

[34] Thouësny, S. 2009. Increasing the Reliability of a Part-of-Speech Tagging Tool for Use with Learner Language. In Automatic Analysis of Learner Language, Automatic Analysis of Learner Language (AALL'09).

[35] Rizzolo, N., Roth, D. 2010. Learning Based Java for Rapid Development of NLP Systems. In LREC.

[36] UOW http://www.uow.edu.au/~dlee/software.htm

[37] Barnickel, T., Weston, J., Collobert, R., Mewes, H. W., Stümpflen, V. 2009. Large Scale Application of Neural Network Based Semantic Role Labeling for Automated Relation Extraction from Biomedical Texts. PLoS One, 4(7), e6393.

[38] Regional real–time transit information system system requirements version 3.0. 2012, http://www.mtc.ca.gov/planning/tcip/Real-Time_TransitSystemRequirements_v3.0.pdf,

[39] Li, W., Brown, D., Hayes, J. H., Truszczynski, M. 2014. Answer-Set Programming in Requirements Engineering. In Requirements Engineering: Foundation for Software Quality p. 168-183. Springer International Publishing.

[40] USEPA. Risk Assessment Guidance for Superfund (RAGS): Volume III — Part A, Process for Conducting Probabilistic Risk Assessment 2001 available at http://www.epa.gov/oswer/riskassessment/rags3adt/pdf/rags3adt_complete.pdf.

[41] Trained Models, http://nilc.icmc.usp.br/nlpnet/models.html

[42] Parse selection for Self-Training, http://depts.washington.edu/uwcl/twiki/pub/Main/JimWhite/Parse_Selection_for_Self-Training.pdf, p. 30