Using the Dependence Level Among Requirements to Priorize the Regression Testing Set and Characterize the Complexity of Requirements Change

André Di Thommazo[1,2], Kamilla Camargo[3], Elis Hernandes[1,2], Gislaine Gonçalves[1], Jefferson Pedro[1], Anderson Belgamo[2,4] 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]*Experteasy, Ribeirão Preto, SP, Brazil*
[4]*IFSP - São Paulo Federal Institute of Education, Science and Technology, Piracicaba, SP, Brazil*

Keywords: Requirements Traceability, Regression Test Prioritization, Requirements Traceability Matrix.

Abstract: Background: *When there are changes in software requirements, other phases of software development are impacted and frequently, extra effort is needed to adjust the previous developed artifacts to new features or changes. However, if the development team has the traceability of requirements, the extra effort could be not an issue. An example is the software quality team, which needs to define effective tests cycles in each software release.* Goal: This papers aims to present an approach based on requirements dependence level to support the regression test prioritization and identify the real impact of requirement changes. Method: The designed approach is based on automatic definition of Requirements Traceability Matrix with three different dependence levels. Moreover, dependence between requirement and test case is also defined. A case study in a real software development industry environment was performed to assess the approach. Results: Identifying the dependence level among requirements have allowed the quality assurance team priorize regression tests and, by means of these tests, defects are early identified if compared with tests execution without priorization. Moreover, the requirements changes complexity is also identified with the approach support. Conclusion: Results shows that definition of dependence levels among requirements gives two contributions: (i) allowing test prioritization definition, which become regression test cycle more effective, (ii) allowing characterize impacts of requirements changes, which is commonly requested by stakeholders.

1 INTRODUCTION

The requirements management is an important activity for the software development process (Cleland-Huang et al., 2012); (Sommerville, 2010); (Zisman and Spanoudakis, 2004). The requirements traceability is the ability to follow the requirement life cycle in both directions: past, related to the features before the inclusion of the requirement, future, related to features after the requirements inclusion (Götel and Finkelstein, 1997).

Requirements management is important because since a requirement is modified or added in a requirements document, all stages of the software development are affected requirements engineering, software modelling, code development and software testing, regardless of the software development process to be followed. To map and manage the requirements linking and other artefacts should be established the requirements traceability. Several authors emphasize the importance of traceability in the software development process as part of requirements management activities (Salem, 2006); (Oliveto et al., 2007).

The requirements traceability is defined as the ability to describe and follow the life of a requirement throughout its life cycle (Guo et al., 2009). This control should cover the whole requirement existence, from its origin - when the requirement is elicited, specified and validated - to the other stages of the software development, including code development and testing phases. Thus, the requirements traceability is a technique that allows the identification and visualization of the dependence between a requirement with other requirements

Di Thommazo A., Camargo K., Hernandes E., Gonçalves G., Pedro J., Belgamo A. and Fabbri S..
Using the Dependence Level Among Requirements to Priorize the Regression Testing Set and Characterize the Complexity of Requirements Change.
DOI: 10.5220/0005468902310241
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 231-241
ISBN: 978-989-758-097-0

and/or other artefacts produced during a software development process.

Cleland-Huang et al., (2012) define two types of traceability:

- *Horizontal*: occurs when there is relationship between requirements and different artefacts such as models, source code and testing artefacts; and
- *Vertical*: when the relationship occurs in the same artefact, for example the relationship between requirements. The dependence between the requirements, in general, is recorded in the Requirements Traceability Matrix (RTM).

In this paper, the vertical traceability is handled by exploiting the level of dependence between the functional requirements (FRs) and the horizontal traceability is handled by using the relationship between each FR and its test cases (TC).

In the context of horizontal traceability, an example is the possibility of identification and prioritization of regression testing from the level of dependence between FRs and between FRs and TCs. According to Rothermel et al., (2001), regression tests are important, but they can also be expensive. In addition, the high cost has motivated several studies to find a better cost effective, especially with the prioritization of TCs (Salem, 2010). As the time devoted to testing activity is limited, prioritization allows to increase the number of serious defects found (Malz et al., 2012).

In the context of vertical traceability, establish the dependence between the Functional Requirements (FRs) can support to predict the impact of changes in software. The analysis of the impact changes allows the project manager to take decisions more effective during the change management of the software development (Kama and Azli, 2012). If a change occurs in a FR with a strong dependence with other requirements, it makes sense to imagine that the maintenance complexity is greater than a FR with no dependence. In this case, the project manager should make some decisions in order to satisfying the stakeholder´s needs. For example, the decision may be related to identifying the level of expertise required to implement the changes.

Despite of the benefits of the requirements traceability, a major challenge for the software development process is to achieve and maintain traceability in a manual way (Sommerville, 2010). Thus, this research group has conducted several studies to propose approaches to automatically generate the RTM. In this sense, the RTM-E and RTM-NLP have been proposed (Di Thommazo et al., 2012). A combination between RTM-E and RTM-NLP approaches enabled the development two new approaches: RTM-N (Di Thommazo et al., 2013a) using neural networks and RTM-Fuzzy (Di Thommazo et al., 2013b) using fuzzy logic. A brief summary of these approaches is presented in Section II. In general, the combination of these approaches allowed the decrease of the number of false positives generated. The problem of false positive is identified in the literature as a typical problem when NLP is used for determining traceability (Sundaram et al., 2010). In addition to reducing the number of false positives, the combination of these approaches indicates the level of dependence between FRs: weak or strong dependence. This feature is a novelty in relation to studies found in the literature. From the dependence level it is possible to generate the TCs prioritization and to identify the change complexity in FRs, as described in Section 5.

Based on the scenario previously presented, this research group developed an approach to define a set of regression tests with prioritized TCs according to the level of dependence between the FRs. Furthermore, we sought to characterize the impact of a change in FRs. This proposal was evaluated by a case study in a real system aiming to show the traceability advantages in order to characterize the impact of changes in the FRs as well as to determine a prioritized set of regression tests with priority.

Thus, the objectives of this paper are:

- To present an approach for selecting a set of regression TCs;
- To show how the level of dependence between FRs in a RTM can be useful in the process of prioritizing a set of regression TCs;
- To show how this novel approach can characterize the impact of changes in FRs, identifying changes with a greater complexity;
- To show the approach application in a real case study.

Note that this paper is not intended to detail the approaches to automatically determine the RTM, since these have been previously published by Di Thommazo et al., (2012), Di Thommazo et al., (2013a), Di Thommazo et al., (2013b) and Di Thommazo et al., (2013c). It also highlights that, despite having several experimental studies to show the effectiveness of these four approaches, for the novel approach, presented in this paper, has not been possible to find a large population such as that undertaken in previous experimental studies. This is because the case study involves monitoring all the software development process, from its requirements specification to software testing, in more than one development cycle, as shown in Section 5.

We also emphasize that there are two systematic

reviews (Singh et al., 2012); (Engström et al., 2010) about the selection of test cases (CTs) for regression testing. It is important to note that in the systematic reviews there were not initiatives that select the Regression Testing Cases with prioritization from the traceability matrix of functional requirements and levels of dependence, as proposed in this paper. In these two studies in the literature are listed techniques to generate Regression Testing with prioritization. One of the techniques cited traceability (Filho et al., 2010) however, this is traceability between models, source code and CTs. Another cited technique (Srikanth et al., 2005) focuses on the requirements, but the definition and prioritization of test cases are not based on traceability. The other techniques mentioned in the work do not have focused on requirements.

This paper is organized as follows: Section 2 presents the definitions of requirements traceability; Session 3 details the regression test concepts and prioritization of TCs; Section 4 shows the proposed approach; Section 5 presents the case study performed; Section 6 presents the findings and future works.

2 REQUIREMENTS TRACEABILITY

The main purpose of requirements management is the requirements organization and storage as well as the management of the changes that occur throughout the development process (Sommerville, 2010); (Zisman and Spanoudakis, 2004). One way to manage the requirements is the establishment of traceability requirements, defined as the ability to describe and monitor the requirements throughout its life cycle (Guo et al., 2009). Generate the link between FRs and the connection between FRs and other artefacts produced is a laborious task of requirements management (Sundaram et al., 2010). The requirements traceability can provide the basis for evolution on requirements changes, in addition to acting directly on the software process quality assurance (Guo et al., 2009). A representation to mapping the dependence between FRs is the creation of RTM, where each FR is represented in one row and one column of the matrix. The dependence between two requirements is recorded in the corresponding cell (line/column intersection). Several authors show the importance of the RTM for the software process development, since the matrix can help predict the impact of a change or of a new FR in the system (Cleland-Huang et al., 2012); (Sommerville, 2010) (Guo et al., 2009); (Goknil et al., 2011.

This research group has proposed four approaches for the automatic generation of RTM. These approaches take into account the FRs of the software, establishing the level of dependence between each pair of FR.

The approaches were developed in COCAR tool (Di Thommazo et al., 2012); (Di Thommazo et al., 2013a); (Di Thommazo, 2013b); (Di Thomazzo et al., 2014), aiming to provide computational support for requirements management process IBM (2012). The COCAR tool uses a template (Kawai, 2005) to store all the requirements. The main purpose of this template is to standardize the data of FRs, avoiding inconsistencies, omissions and ambiguities.

Thus, the four approaches are available in the COCAR tool, aiming to generating the RTM:

- RTM-E: based on input data from FRs and explained in Di Thommazo et al., (2012);
- RTM-NLP: based on natural language processing (NLP) and explained in Di Thommazo et al. (2012);
- RTM-Fuzzy: combines the RTM-E and RTM-NLP approaches using fuzzy logic (Di Thommazo et al., 2013b). The pertinence functions of the fuzzy system were defined by genetic algorithms, as described in Di Thommazo et al., (2014);
- RTM-N: combines the RTM-E and RTM-NLP approaches using neural networks (Di Thommazo et al., 2013a).

In COCAR tool is also possible to register TCs and associated them to FRs previously recorded. This feature supports the vertical traceability.

3 REGRESSION TEST AND TEST CASE PRIORITIZATION

The process of verifying the modified software in maintenance phase is called Regression Test (Maheswari and JeyaMala, 2013). The regression test should be performed after the software suffer any change, either by a new requirement or changes in an existing requirement, in order to validate if those modifications do not have inserted new faults. The number of test cases for regression test have a continuously grow whilst software is developed, and re-execute all of them is not always feasible due to schedule constraints (Kukreja et al., 2013).

The regression test is an expensive phase of software testing process, a modification in the software can require re-execute all set of test cases for a module, for example. If the regression test suite is

too large more effort will be required. The execution of a regression test requires a plan, which includes planning how, who and when will be executed (Myers et al., 2004). The planning allows a better efficiency in this activity.

Researchers have considered various techniques to reduce costs of regression tests between them: test case selection and test suite minimization (Rothermel et al., 2001). The test case prioritization is a technique that helps the identification of a subset of test cases that will require less effort and time to be executed during regression test phase. This subset, which comprises test cases prioritized, helps the team in early achievement of goals of the tests (Maheswari and JeyaMala, 2013). In this technique the team will sort the test cases according with his priority, following a selection criterion. Test cases with lower priority will be executed after the test cases with higher priority (Rothermel et al., 2001). This can increase the probability of finding faults early during test process. As soon the faults are revealed faster the software can be fixed, which means less time to product delivery. The test case prioritization should follow a clear criteria that allows identify why a test case has higher priority than other. This criteria need to be aligned with the expected goal to be achieved by decreasing the set of regression tests: costs reduction, increase fault detection rate, reduces product delivery time, between others.

The next section presents the proposed approach, which use the dependence level between functional requirements as criteria for the test case prioritization.

4 PROPOSAL APPROACH

In this section we present the proposed approach to address the changes in FRs during the software development process. It is noteworthy that when we mention changes in requirements, we are referring to the modification of some existing functionality already implemented or the inclusion of a new FR. The objective of this proposal is to select a set of regression TCs, by means of prioritization, from any modifications in a FR or in a set of FRs. The approach was based on the level of dependence between FRs during the RTM generation.

Thus, once the system FRs are specified and associated with their respective TCs, the regression tests executed after any FR modification/inclusion are prioritized in order to show different alternatives of testing execution. As previously mentioned, the COCAR tool was used to support the proposed approach by means of FRs registration, RTM generation (Di Thommazo et al., 2012); (Di Thommazo et al., 2013a); (Di Thommazo et al., 2013b); (Di Thommazo et al., 2014) and the registration of TCs for each FR.

To illustrate the approach, consider a software that was developed and its FRs and TCs have already registered in COCAR tool. Any software modification should be realized by means of FR modification, codification and testing. Figure 1 shows the flowchart of the proposed approach. Note that the result of the approach is that the set of regression tests have priorities according to the level of dependence between the FRs. The higher the priority of the TC, more chances it will have to find a defect and, therefore, must be run first (Rothermel et al., 2001).

In addition to setting the TCs priority, this approach allows to characterize the impact of changes in the software. Therefore, the development team should know and convey the complexity related to each of the changes before the implementation. Based on this information can be taken actions for risk management, aiming to create contingency plans for great complexity changes, or even to decide the best time to implement complex changes. It may also be possible to allocate more experienced staff members to deal with more complex problems. It is noteworthy that this approach does not define a metric to measure the impact of changes related to effort in person-hour or function points. The idea is to present a comparison among the changes, so you can characterize among all of them, which one presents the highest risk due to the highest dependence among the FRs. As previously mentioned, the definition of regression tests is based on the dependence among the FRs and between the FRs and the TCs. Therefore, we used the same dependencies to characterize the complexity of the changes, assuming that the greater the dependence among the FRs, the harder it is to be maintained.

If a FR has dependence with several others, any change in this FR implies to adjust models with strong dependence with another models and source-code snippets with strong dependence with other features. Thus, the understanding of other code snippets are necessary to make the change in the FR. In general, the effort to understand third-party source code involves a considerable time and attention from developers. Since this is a possible source of defects, the testing and correction effort may be longer. Moreover, to modify or insert a FR, that has a weak dependence (or no dependence) with others FRs implemented, does not present difficulty of adjusting to implement other functions already created. To characterize the impact of change, we used the amount of TCs that must be performed for each

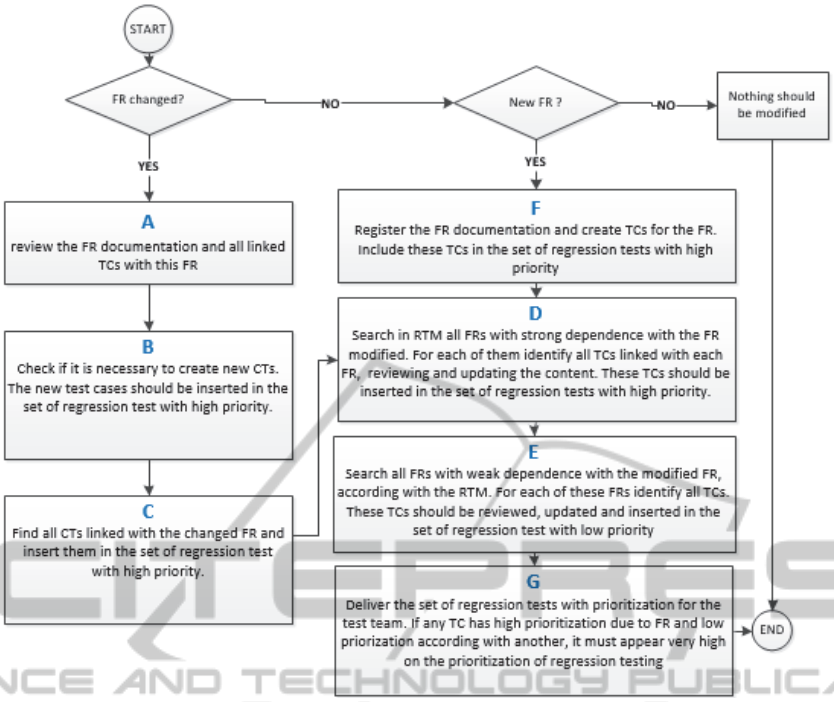

Figure 1: Steps to select prioritized TCs based on the dependence of FRs.

change, considering their respective priorities. A simple counting of FRs with dependencies should not be realized because if a FR has many TCs, it may indicate that one FR has a complex business rule. On the other hand, if a FR has only one associated TC, it may indicate that your business rule is simple.

Thus, to characterize the impact of changes we took into account the priority of the TCs, setting weight 3 for high priority and weight 1 for low priority. Consider the priority of TC is the same the dependence level of the FRs. Equation 1 shows how it is done the calculation of this complexity. The weight values were defined considering that a complex change can be 3 times harder than a simple change. It is understood that this value can be adjusted based on other data being collected by the project manager. It is noteworthy that, in this study, these values were consensus on the development team.

$$C_{FRx} = (\text{qtdPrHigh} * 3) + (\text{qtdPrLow}) \quad (1)$$

In Equation 1, qtdPrHigh is the amount of TCs with high priority identified by the approach to change the FRx and qtdPrLow is the amount of TCs with slow priority identified by the approach to chnge the FRx. Figure 2 summarizes the approach for the complexity characterization.

In the next section we present the case study of a professional system, whose evolution and changes in the FRs followed this approach.

5 CASE STUDY

This case study was conducted to evaluate the set of regression testing with test cases prioritized based on the level of dependence between FRs and traceability between FR and TC. The case study also characterizes complexity for the modification of a FR, also using the level of dependence between FRs (as defined in the RTM) and regression tests that were determined to test the requested change.

The software that was developed during the case study has the target of evaluating physical activity of patients undergoing step test efforts. The software was part of a project of the Department of Physiotherapy at UFSCar and is linked with a patent for a new product. During the evolution of this product several changes had been necessary in the software. In this iterative and interactive scenario changes, the ability to deal with changes in requirements assists the product development process. When we talk about selection of CTs with prioritization, we are referring to the ability to set the priority of each CT that will identify more defects and faster, in addition to covering the changes that occurred in the software. The steps performed in the case study were:

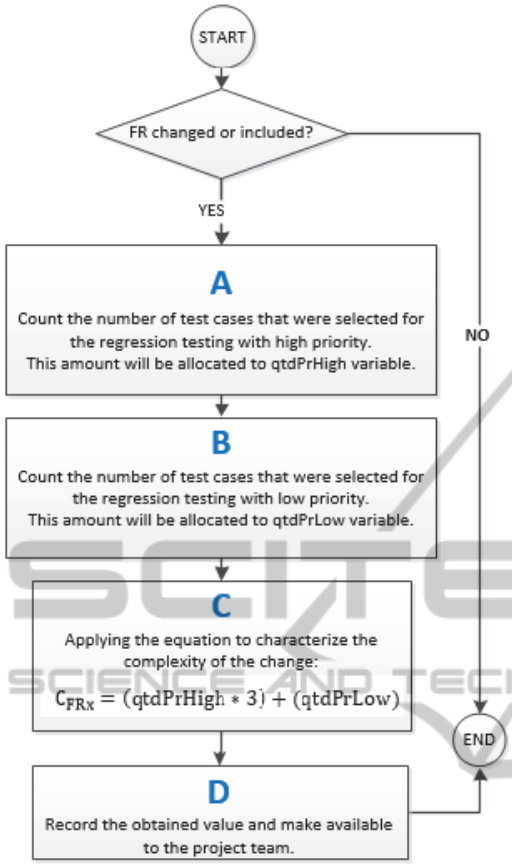

Figure 2: Proposed approach to characterize the complexity of changes in functional requirements.

Step 1) *Record of Functional Requirements in COCAR Tool:* requirements were elicited with people who would be users of the system. All requirements were registered in COCAR tool, by adding the description, processing, data entry, constraints, outputs and stakeholders involved;

Step 2) *Generation RTM According to the RTM-Fuzzy Approach:* Figure 3 illustrates the RTM generated. Green color cells indicate a weak dependence between FRs. Red color cells indicate a strong dependence between FRs. The RTM-Fuzzy approach was detailed in Di Thommazo et al. (2013b);

Step 3) *Insertion of TCs in COCAR Tool:* from the analysis of each FR were created functional test cases related with each requirement. As the TCs were created from the FRs, traceability between FR and TCs was established automatically in the tool. This traceability is required for identifying the priority of test cases. If a FR is changed then it is necessary execute first all TCs linked to this FR in regression test phase, as detailed in the sequence;

Step 4) *Software Development:* the development process involved two developers and was developed with C # language to desktop environment.

Step 5) *Test Case Execution of to find Bugs in the Software Developed:* the test cases produced on step 3 were executed after completion of software development phase. The errors found were fixed and the TCs that raised defects were executed again. The execution of 106 test cases spent about 160 minutes.

After these steps the software was delivered to the Department of Physiotherapy for the acceptance test phase. The software was used in day to day evaluation of patients undergoing physical stress tests. It was expected that after the first execution of the acceptance test, improvements were requested in the proposed software: both as regards the change in the existing FRs and the insertion of new FRs. In addition, it was expected that defects could be found which were not detected in the first test phase.

Step 6) *Changes Treatment:* after two weeks of use,

	RF01	RF02	RF03	RF04	RF05	RF06	RF07	RF08	RF09	RF10	RF11	RF12	RF13	RF14	RF15	RF16	RF17	RF18	RF19	RF20	RF21	RF22	RF23	RF24	RF25	RF26	RF27	RF28	RF29	RF30
RF01	-	100.0...	66.67%	66.67%	16.67%	16.67%	20.00%	25.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	9.09%	0.00%	0.00%	0.00%	0.00%
RF02	100.0...	-	66.67%	66.67%	16.67%	16.67%	20.00%	25.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	9.09%	0.00%	0.00%	0.00%	0.00%
RF03	66.67%	66.67%	-	100.0...	20.00%	20.00%	25.00%	33.33%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%
RF04	66.67%	66.67%	100.0...	-	20.00%	20.00%	25.00%	33.33%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	0.00%	0.00%	0.00%	0.00%
RF05	16.67%	16.67%	20.00%	20.00%	-	100.0...	75.00%	50.00%	5.00%	5.00%	0.00%	20.00%	0.00%	0.00%	0.00%	0.00%	16.67%	16.67%	20.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	18.18%	0.00%	0.00%	0.00%	0.00%
RF06	16.67%	16.67%	20.00%	20.00%	100.0...	-	75.00%	50.00%	5.00%	5.00%	0.00%	20.00%	0.00%	0.00%	0.00%	0.00%	16.67%	16.67%	20.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	18.18%	0.00%	0.00%	0.00%	0.00%
RF07	20.00%	20.00%	25.00%	25.00%	75.00%	75.00%	-	66.67%	5.26%	5.26%	0.00%	25.00%	0.00%	0.00%	0.00%	0.00%	20.00%	20.00%	25.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	20.00%	0.00%	0.00%	0.00%	0.00%
RF08	25.00%	25.00%	33.33%	33.33%	50.00%	50.00%	66.67%	-	5.56%	5.56%	0.00%	33.33%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF09	0.00%	0.00%	0.00%	0.00%	5.00%	5.00%	5.26%	5.56%	-	100.0...	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF10	0.00%	0.00%	0.00%	0.00%	5.00%	5.00%	5.26%	5.56%	100.0...	-	88.24%	11.76%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	5.56%	5.56%	5.88%	5.88%	5.56%	8.33%	0.00%	11.76%	11.76%	11.76%
RF11	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	88.24%	-	6.25%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	6.25%	6.25%	6.67%	6.67%	6.25%	4.35%	0.00%	13.33%	13.33%	13.33%
RF12	0.00%	0.00%	0.00%	0.00%	20.00%	20.00%	25.00%	33.33%	11.76%	11.76%	6.25%	-	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	33.33%	22.22%	0.00%	0.00%	0.00%	0.00%
RF13	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	-	100.0...	66.67%	66.67%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF14	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.0...	-	66.67%	66.67%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF15	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	66.67%	66.67%	-	100.0...	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF16	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	66.67%	66.67%	100.0...	-	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF17	0.00%	0.00%	0.00%	0.00%	16.67%	16.67%	20.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	-	100.0...	66.67%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF18	0.00%	0.00%	0.00%	0.00%	16.67%	16.67%	20.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.0...	-	66.67%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF19	0.00%	0.00%	0.00%	0.00%	20.00%	20.00%	25.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	66.67%	66.67%	-	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF20	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	-	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
RF21	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	5.56%	6.25%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	-	100.0...	50.00%	50.00%	0.00%	0.00%	0.00%	33.33%	33.33%	33.33%
RF22	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	5.56%	6.25%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.0...	-	50.00%	50.00%	0.00%	0.00%	0.00%	33.33%	33.33%	33.33%
RF23	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	5.88%	6.67%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	50.00%	50.00%	-	100.0...	0.00%	0.00%	0.00%	50.00%	50.00%	50.00%
RF24	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	5.88%	6.67%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	50.00%	50.00%	100.0...	-	0.00%	0.00%	0.00%	50.00%	50.00%	50.00%
RF25	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	5.56%	6.25%	33.33%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	-	10.00%	0.00%	0.00%	0.00%	0.00%
RF26	9.09%	9.09%	10.00%	10.00%	18.18%	18.18%	20.00%	0.00%	0.00%	8.33%	4.35%	22.22%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.00%	-	0.00%	0.00%	0.00%	0.00%
RF27	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	-	0.00%	0.00%	0.00%
RF28	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	11.76%	13.33%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	33.33%	33.33%	50.00%	50.00%	0.00%	0.00%	0.00%	-	100.0...	100.0...
RF29	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	11.76%	13.33%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	33.33%	33.33%	50.00%	50.00%	0.00%	0.00%	0.00%	100.0...	-	100.0...
RF30	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	11.76%	13.33%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	33.33%	33.33%	50.00%	50.00%	0.00%	0.00%	0.00%	100.0...	100.0...	-

Figure 3: RTM obtained from the system of case study, before the proposed changes.

the development team met with users to evaluate the software. The following requests for modification were made:

- Change in FR9 related to the patient record, inserting a new field;
- New FR related to the inclusion of the completion of a questionnaire by patients (will be called FR31);
- New FR related to the system backups (will be called FR32).

The steps used to determine the implementation of these modifications are described in Figure 1. A detailed description of each step performed in this case study is presented following.

To deal with the change in FR9 it was necessary to review the documentation relating to all TCs that are related with this FR (Figure 1 step A). When an FR is modified, the CTs related to it should be reviewed and updated to continue according to the FR. After an analysis was made to check if you need to create some new TC, which did not occur for the requirement in question, the FR9 (Figure 1 step B).

Following it was identified all TCs related FR9. The selected TCs were TC34, TC35, TC36 and TC37 (Figure 4 step A). These four TCs were inserted in the regression test set with high priority, according with the step C of Figure 1.

The next step was to identify all FRs that had strong dependence with FR9 (according to the RTM-Fuzzy approach (Di Thommazo et al., 2013b). The only FR that had a strong dependence was the FR10. This requirement is related to the update of a patient in the system. This relationship is shown by the red arrow in Figure 4. All the FR10 TCs were entered into the regression test universe with high priority (TC38, TC39 and TC40). It was necessary to create a new TC, called TC107, so that the new inserted field could be validated. This TC was also inserted into the regression test universe with high priority, according the Figure 1 step 4 and represented by Figure 4-B.

Finally, were recovered the FRs which had low dependence with the FR9 (FR5, FR6, FR7 and Fr8) labelled with green arrow in Figure 4 and their respective TCs (TC19 to TC33) as described in Figure 1 step E. These TCs were added to regression test suite with low priority, as shown in Figure 4-C.

The last step of the proposed tests would be provide for the test team the TCs from TC34 to TC40 with high priority and the TC23 to TC33 with low priority (Figure 1 step F).

Another change in the set of requirements was the insertion of the FR31, related to the completion of a survey by patients. After the inclusion of this requirement more TCs were created for this functionality (TC108, TC109 and TC110). They were added to the regression test suite with high priority (related to results of Figure 1 step F and shown in Figure 5-A).

Following the proposed approach, we find out in the RTM generated by the RTM-Fuzzy approach all FRs with strong dependence with the FR31. In this case, there was no FR in this situation (Figure 1 step D).

The next step (Figure 1 step E) was identified which FRs have weak dependence with the FR that was inserted (FR31). In this case were recovered FR9, FR10, FR11, FR12, FR25 and FR26. These weak dependencies are marked in Figure 5 by the green arrows. The TCs related to these six FRs were recovered and inserted into the set of regression TCs with low priority. These TCs recovered from the insertion of the FR31 were made available to the test team (Figure 1 step G).

Finally, we detail the insertion of the FR32, related with the creation of import system backups. As this is a new FR, the first action was to create the TCs for this FR (Figure 1 step F shown in Figure 6). The next step was to search the FRs with strong dependence and then select the TCs connected to them (Figure 1 step D). There were no FRs with strong or weak dependence (Figure 1 step E). Thus, only the TCs created specifically to address this requirement (FR32) were inserted into the set of regression TCs.

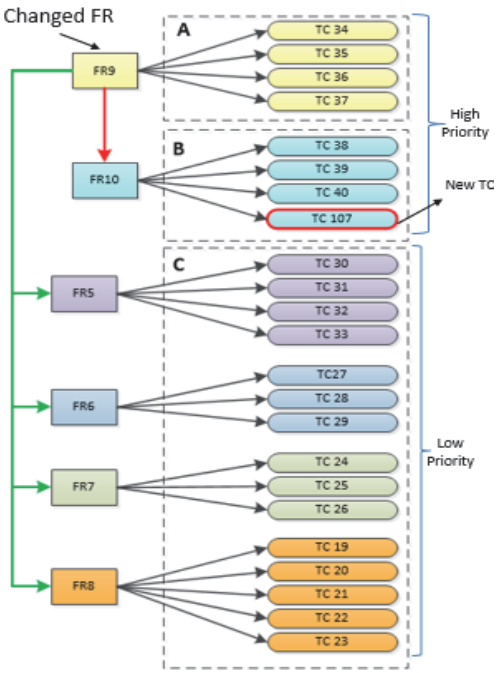

Figure 4: Changes in FR9.

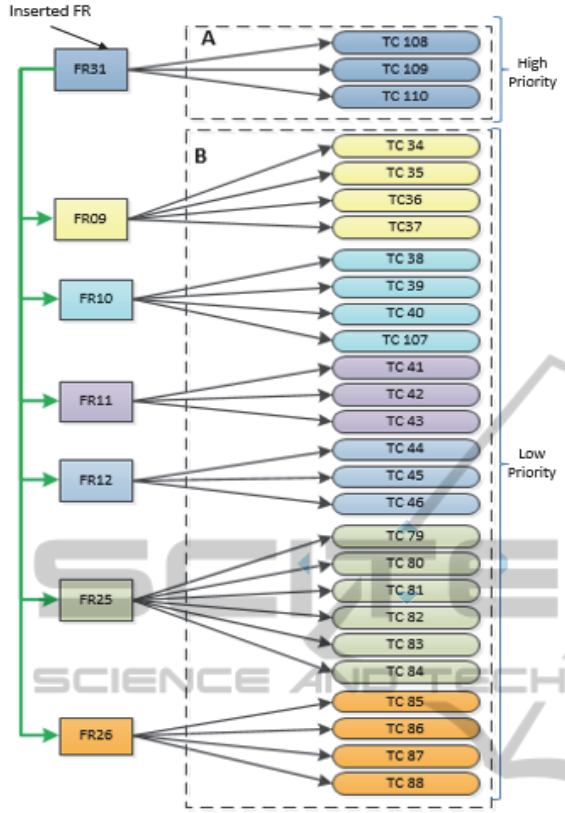

Figure 5: Insertion of the FR31.

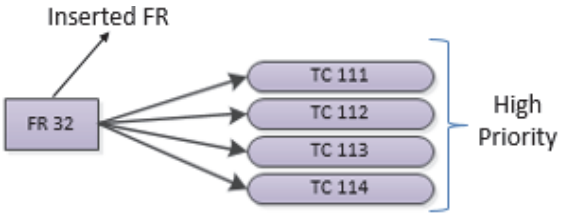

Figure 6: Insertion of the FR32.

The result of the execution of the steps described in Figure 1 was a set of regression tests. Note that the TCs TC34, TC35, TC36 and TC37 were given high priority when the FR9 was modified and low priority when the FR31 was inserted. In these situations should always be considered the highest priority, as detailed in Figure 1 step G.

Step 7) *Characterization of the Impact of Changes:* to illustrate the characterization of complexity from the approach, Table 1 was built with case study data. Equation 1 was used to characterize the complexity of each change in the FRs. In the following (Equation 2) is an example of the calculation was made for the change in FR9:

$$C_{RF9} = 8 * 3 + 15 = 39 \quad (2)$$

The same procedure was used to describe the complexity of FR31 and FR32 requirements. It is possible to realize that the change in FR9 and FR31 requirements has a greater impact than the FR32. This information can help the project manager to allocate more experienced people for more complex tasks. Despite, when there are several changes with great complexity, this information can helps the project manager estimate what will be the time for a new software release. With time and knowledge of the company under its own process is possible to trace the relationship of the absolute value of the complex with man-hour efforts or function points that was made to calculate the change in FR9.

Step 8) *Implementation of the Changes in the Software: t*he change in FR9 and insertion of FR31 and FR32 were implemented by the development team;

Step 9) *Application of Regression Testing for Defects Detection:* in this step, the set of regression tests with priority was performed in software built in the previous step (step 7). This set of TCs had two categories:

Table 1: Complexity of the FRs modifications.

FR	TCs from FR	TCs that must be performed with high priority	TCs that must be performed with low priority	TCs quantity with high priority (qtdPrHigh)	TCs quantity with low priority (qtdPrLow)	Characterization of the complexity to modify the FR
FR9	TCTC34, TC35, TC36, TC37	TC34, TC35, TC36, TC37, TC38, TC39, TC40, TC107	TC19, TC20, TC21, TC22, TC23, TC24, TC25, TC26,TC27, TC28, TC29, TC30	8	15	39
FR31	TC108, TC109, TC110	TC108, TC109, TC110	TC34, TC35, TC36, TC37, TC38, TC39, TC40, TC107, TC41, TC42, TC43, TC44, TC45, TC46, TC79, TC80, TC81, TC82, TC83, TC84, TC85, TC86, TC87, TC88	3	25	34
FR32	TC111, TC112, TC113	TC111, TC112, TC113, TC114	-	4	0	12

- 15 TCs with high priority: TC34, TC35, TC36, TC37, TC38, TC39, TC40, TC107, TC108, TC109, TC110, TC111, TC112, TC113 and TC114;
- 28 TCs with low priority: TC19, TC20, TC21, TC22, TC23, TC24, TC25, TC26, TC27, TC28, TC29, TC30, TC41, TC42, TC43, TC44, TC45, TC46, TC79, TC80, TC81, TC82, TC83, TC84, TC85, TC86, TC87 and TC88.

The execution of High Priority tests had the following results: 6 cases failed and 9 passed. The TCs that failed were: TC37, TC40, TC107, TC112, TC113 and TC114. Approximately 20 minutes was spent to perform these 15 tests. After running the Low Priority tests following TCs failed: TC42, TC44, TC84. Approximately 35 minutes was spent to perform these 28 tests.

In order to measure the difference between the performances of all 114 TCs without the prioritization the tests were executed again, with the same version of software built in step 8. The TCs were performed in ascending order (the TC01 to the TC114). It was spent about 160 minutes and the same 9 tests that failed in the previous step failed. Table 2 summarizes the data of this step.

Step 10) *Evaluate the Effectiveness of Regression Testing:* based on the data of Table 2 it can be seen that:

- The time to perform regression testing was approximately three times shorter than the time used to perform all the tests (50 minutes versus 160 minutes);
- All defects found during the execution of the entire test suite were also found in the set of priority regression tests;
- The high priority TCs detected more defects compared to low priority TCs (6 versus 3).

The most critical defects were found first, which shows that the test case prioritization can increases the defect detection rate. This allows that the most serious defects that are early identified can be fixed first. In a test design, with enough time constraints for regression test or limited budget, prioritization helps ensure the quality of the product even after such a modification there is not enough time to perform the entire set of tests during test regression.

As threats to validity of the case study can be highlighted the fact that the test team did not have great practical experience in software testing. The test team received a brief conceptual training before the start of activities. It also highlights the fact that although generate a product to market, there were few involved people in software development: two analysts, two software developers and a tester.

Table 2: Case study data summarization.

		# of TCs		Time spent (min)		Raised defects	
Regression Tests	High priority	15	40	20	50	6	9
	Low priority	25		30		3	
TCs of all tests cycles		114		160		9	

6 CONCLUSIONS AND FURTHER ACTIONS

This paper presented an approach to select TCs regression with prioritization and characterize the complexity of changes in FRs. This approach was based on the dependence levels between FRs defined in RTM. The evaluation approach was taken in a real case study.

The identification of dependence levels between FRs allowed the identification of the priority of each TC and characterization of the complexity of the changes in requirements. The results of the case study shows that the approach reduced the set of tests with the same efficacy as if performed the entire test suite in the software maintenance phase (regression tests). The reduced set to 9 TC found the same defects, but approximately three times faster. This indicates that the prioritization of TCs become more efficient then execution of all TCs. The identification of the TCs that should be part of Regression Test with the priorization are defined automatically by COCAR tool after a FR to be inserted or modified.

Regarding the characterization of the complexity of the changes the approach estimated the changes that would be more complex to be performed in software. We cannot ensure a direct relation between the complexity and the effort to implement them. Further studies are planned to assess the results of this approach. Despite the characterization of complexity make sense for the team (analysts, developers and testers) it needs to be investigated and measured by other studies that will be conducted by the group.

As future work for the selection of regression tests with prioritization, we intend to use in addition of the relation between each other FRs in the RTM and the relationship between FRs and the TCS, the relationship between the TCs. Therefore, we can improve the selection of regression and prioritizing

tests. Currently is being implemented in the COCAR tool a module that deals with the definition of that traceability between TCs for the refinement of the approach and conducting new case studies. The private aviation company that work as a partner of the research group in the work reported in Di Thommazo et al., (2012) and Di Thommazo et al., (2013a) has prepared the environment and engaged to conduct the new case study.

During the execution of this work was also realized that it is possible to map the relationship between the complexity of the stress changes to metrics such as man/hour, and metrics related to the software size, for example, function points. The same company previously mentioned has a good historical basis, whose data can confirm this characterization.

REFERENCES

Cleland-Huang, J., Gotel, O., Zisman, A. (2012) *Software and Systems Traceability*, Springer, Berlin.

Di Thommazo, A., Malimpensa, G., Olivatto, G. , Ribeiro, T., Fabbri, S. (2012). Requirements Traceability Matrix: Automatic Generation and Visualization. In: *Brazilian Symposium on Software Engineering*, SBES, Natal, Brazil, Sep. 2012. Los Alamitos: IEEE Press.

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

Di Thommazo, A., Ribeiro, T., Olivatto, G., Werneck, V., Fabbri, S. (2013b) An automatic approach to detect traceability links using fuzzy logic. In: *Brazilian Symposium on Software Engineering*, SBES, Brasília, Brazil, Sep. 2013. Los Alamitos: IEEE Press.

Di Thommazo, A., Rovina, R., Ribeiro, T., Olivatto, G., Hernandes, E., Werneck, V., Fabbri, S. (2014) Using artificial intelligence techniques to enhance traceability links. In: *International Conference on Enterprise Information Systems*, ICEIS, Lisboa, Portugal. April 2013. Lisboa: INSTIC Press.

Engström et al. A Systematic Review on Regression Test Selection Techniques (2010) Information and Software Technology, 52, 1, 2010, p. 14–30

Filho et al. Supporting concern based regression testing and prioritization in a model driven environment (2010). In: Annual Computer Software and Applications Conference Workshop, COMPSAC, Seul, July 2010. New York: ACM Press.

Goknil, A., Kurtev, I., Van den Berg, K., Veldhuis, J. W. (2011) Semantics of Trace Relations in Requirements Models for Consistency Checking and Inferencing. *Software and Systems Modeling*, 10, 1, Feb. 2011.

Götel, O., Finkelstein, A. (1997) Extended requirements traceability: results of an industrial case study In: Third IEEE International Symposium on Requirements Engineering, Annapolis, USA

Guo, Y., Yang, M, Wang, J., Yang, P., Li, F. (2009) An Ontology based Improved Software Requirement Traceability Matrix. In: *International Symposium on Knowledge Acquisition and Modeling*, KAM , Wuhan, China, Nov. 2009, Los Alamitos: IEEE Press.

IBM. (2012) *Ten Steps to Better Requirements Management*. Available at: http://public.dhe.ibm.com/common/ssi/ecm/en/raw14059usen/RAW14059USEN.PDF. (Accessed: 18 January 2015).

Kawai, K. K. (2005) *Guidelines for preparation of requirements document with emphasis on the Functional Requirements* (*in portuguese*). Master in Computer Science, Federal University of São Carlos, Brasil, 170 f.

Kama, N.; Azli, F. (2012) A Change Impact Analysis Approach for the Software Development Phase. In: Asia-Pacific Software Engineering Conference, APSEC, Hong Kong, Dec. 2012. New York: ACM Press.

Kukreja, N.; Halfond, W.G.J.; Tambe, M. (2013) Randomizing regression tests using game theory. In: *IEEE/ACM International Conference on Automated Software Engineering*, ASE, San Francisco, USA, Nov. 2013, Los Alamitos: IEEE Press.

Maheswari, R. U.; JeyaMala, D., "A novel approach for test case prioritization. (2013). In: *IEEE International Conference on Computational Intelligence and Computing Research*, ICCIC, Tamilnadu, India, Dec. 2013, Los Alamitos: IEEE Press.

Malz, C.; Jazdi, N.; Gohner, P. (2012) Prioritization of Test Cases Using Software Agents and Fuzzy Logic. In: IEEE Software Testing, Verification and Validation, ICST, Montreal, Canada, Apr. 2012, Los Alamitos: IEEE Press.

Myers, G. J. et al. (2004) *The art of software testing*. Nova Jersey: John Wiley and Sons, 2004.

Oliveto, R.; Antoniol, G.; Marcus, A.; Hayes, J. (2007) Software Artefact Traceability: the Never-Ending Challenge. In: *IEEE International Conference on Software Maintenance*, ICSM, Paris, Oct. 2007, Los Alamitos: IEEE Press, pp.485,488.

Rothermel, G.; UnCTh, R.H.; Chengyun Chu; Harrold, M.J. (2001) Prioritizing test cases for regression testing. *IEEE Transactions on Software Engineering*, 27, 10. October 2001. pp.929,948

Sundaram, S.K.A., Hayes, J.H.B., Dekhtyar, A.C., Holbrook, E.A.D. (2010) Assessing Traceability of Software Engineering Artifacts. In" *International IEEE Requirements Engineering Conference*, Sydney, Australia, Sep. 2010, Los Alamitos: IEEE Press.

Salem, A. M. (2006) Improving Software Quality through Requirements Traceability Models. In: *ACS/IEEE Int. Conf. Computer Systems and Applications*, AICCSA, Dubai, Sharjah, March 2006. Los Alamitos: IEEE Press.

Salem, Y. I.; Hassan, R. (2010) Requirement-based test case generation and prioritization. In: Computer

Engineering Conference, ICENCO, Giza, Egypt, Dec, 2010, Los Alamitos: IEEE Press.

Singh et al. Systematic Literature Review on Regression Test Prioritization Techniques (2012). Informatica, 36, 2012, p. 379–408

Sommerville, I. (2010) *Software Engineering*. Addison Wesley, New York, 9th edition

Srikanth, H.; Williams, L.; Osborne, J. (2005) System test case prioritization of new and regression test cases. In: International Symposium on Empirical Software Engineering, Noosa Heads, Australia, November 2005. Los Alamitos, IEEE Press.

Zisman, A., Spanoudakis, G. (2004) Software Traceability: Past, Present, and Future. *The Newsletter of the Requirements Engineering Specialist Group of the British Computer Society*, September 2004.