Regression Testing Prioritization Based on Fuzzy Inference Systems

Pedro Santos Neto - UFPI - Teresina - PI - Brazil
pasn@ufpi.edu.br
Ricardo Britto - UFPI - Teresina - PI - Brazil
pasn@ufpi.edu.br
Thiago Soares - UFPI - Teresina - PI - Brazil
thiagoacs2@gmail.com
Werney Ayala - UFPI - Teresina - PI - Brazil
werney@gmail.com
Jonathas Cruz - UFPI - Teresina - PI - Brazil
jonathas_jivago@hotmail.com
Ricardo Rabelo - USP - São Paulo - SP - Brazil
ricardor@sc.usp.br

Abstract—The software testing is a fundamental activity related to product quality. However, it is not performed in suitable way by many organizations. It is necessary to execute testing in a systematic and planned way. This work presents a fuzzy inference system for test case prioritization, based on the use of inputs related to volatility, complexity and relevance of requirements. The developed inference system allows the specification of a prioritization strategy by the tester based on linguistic rules in a simple and easy way.

I. INTRODUCTION

Testing is the final verification of a software product [1]. Despite of the relevance of this phase on a project, testing is not systematic fulfilled for the enterprises. That situation happens mainly because of the costs and the short schedules related to the software projects.

The software engineering community must find ways to facilitate the inclusion of systematic testing into software processes. In some enterprise environments, which use software testing to assure the software quality, might be necessary prioritize existing test cases because often there is not enough time or resources to execute all planned test cases. In this case, it is desirable to prioritize the test cases, in order to maintain the most important ones in the first positions, assuring their execution. To do that, it is necessary create a execution order that optimizes the most important features for the testing in a certain time, such as code covering.

Since creating the test execution order is a np-hard problem [2], it is not possible to solve that problem in an efficient way through standard techniques [3]. Thus, many researchers have applied computational intelligence techniques to solve the test case prioritization problem ([4], [5], [6]).

This work presents an approach based on Mamdani fuzzy inference system [7] that uses requirements to solve the test case prioritization problem ([8], [5], [9]). The approach allows that a test engineer can insert his knowledge about the prioritization process into the fuzzy inference system. Therefore, the fuzzy inference system mimics the behavior of the expert.

The main contribution of this work is the use of a fuzzy inference system to qualify each existing test case of a software project. This feature allows the prioritization through a simple sort algorithm, driven by the value inferred for each test case, based on common input data as relevance of the requirement for the project, its complexity and volatility. The complexity and volatility can be easily obtained from common software metric tools, like Sonar¹.

This work is organized as following: in Section 2 is presented a software testing primer; in Section 3 a fuzzy inference system is described; in Section 4 the proposed approach is detailed; in Section 5 an empirical evaluation is presented; in Section 6 some related works are discussed; finally, in Section 7 the conclusions and future works are presented.

II. SOFTWARE TESTING PRIMER

Software Testing is one of the main activities of quality assurance in software development. Software testing consists of the dynamic verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior.

There are different classifications related to software testing. Tests can be classified by the target: one module (unit), several modules grouped (integration) and all the modules together (system).

Testing can be aimed at verifying different properties. This represents the test classification by objectives. Test cases can be designed to check that the functional specifications are correctly implemented (functional testing). Other important objectives for testing include (but are not limited to) reliability

¹http://www.sonarsource.org/
measurement, usability evaluation, performance, stress and acceptance, for which different approaches would be taken. Note that the test objective varies with the test target; in general, different purposes being addressed at a different level of testing.

According to IEEE [10], regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects. In practice, the idea is to show that software did not regress, that is, the software which previously passed the tests still does. The repetition of tests is intended to show that the software's behavior is unchanged, except insofar as required. Obviously a trade-off must be made between the assurance given by regression testing every time a change is made and the resources required to do that. A important limiting factor applied to regression testing is related to time-constraints. Sometimes, the total execution of testing can take many hours. This makes unfeasible the test execution constantly. Due to this factor it is required to prioritize the test cases, in order to create the best sequence that meets the time constraints.

It is relevant to emphasize that the selection must be found quickly and the process to find this must not require the tests execution, since the objective is right to reduce the test execution.

Unfortunately, several approaches developed so far, mainly related to the use of search based techniques applied to this problem, are only theoretical proposals, that can not be executed in a real software development environment.

It is mandatory the use of information that can be easily obtained by automatic tools. Therefore, it is mandatory the use of algorithms that do not require the tests execution to find the best order. The sequence must be inferred from test properties.

III. FUZZY INFERENCE SYSTEMS

Fuzzy inference systems are based on linguistic production rules like “if ... then”. The fuzzy sets theory [11] and fuzzy logic [12] provide the math base required to deal with complex processes, based on inaccurate, uncertain and qualitative information.

Fuzzy inference systems have their operation based on three steps: i) fuzzification, ii) inference procedures, and iii) defuzzification. The fuzzification is a mapping of the numerical inputs to fuzzy sets. Those sets are represented by linguistic terms, such as "very", "little", "medium", etc. The fuzzy inference procedure is responsible to infer output fuzzified value based on the input values. Defuzzification is applied to associate a numerical value to a fuzzy output, obtained from the fuzzy inference procedure.

In this work, we applied a Mamdani fuzzy inference system model. That system model was applied because it computationally simulate the human ability to take rational decisions in an environment with inaccuracies, uncertainties and noise [7]. Production rules in a Mamdani inference model have fuzzy sets in their antecedents and consequents. Therefore, a rule base in a Mamdani model can be defined exclusively in a linguistic way, without the need of numerical data.

A fuzzy inference systems can express and handle qualitative information. This allows the mapping of the experience of domain specialists, facilitating the decision making process. Thus, using a Mamdani fuzzy inference system, to solve the test case prioritization problem, allows to get an action strategy / control that can be monitored and interpreted from the linguistic point of view. Thus, the action / control strategy of the Mamdani fuzzy inference system can be considered as reasoned and consistent as the strategy of domain experts.

IV. PROPOSED APPROACH

Our approach evaluates the requirements to infer the test case criticality for regression testing purpose. However, test cases from real systems can eventually cover artifacts that may be related to more than one requirement. In the current stage of our work we do not take into account this behavior.

Each requirement is evaluated based on 3 variables that represent the volatility, complexity and relevance. By using these variables a value that represents the criticality of a test case is generated. We projected a Mamdani fuzzy inference system to infer the test case criticality, as shown in Figure 1.

The crisp input variables of the fuzzy inference system are Volatility, Complexity and Relevance. The meaning of each variable is described as following:

- Volatility (V): value representing the amount of versions of the artifacts related to a requirement. The amount of versions can be easily obtained from a control version systems like GIT² and SVN³.
- Complexity (C): value representing the cyclomatic complexity of the artifacts related to a requirement. This can be easily obtained from a common software metrics tools, like Sonar.
- Relevance (R): value representing the relevance of the requirement for the customer.

The values related to volatility and complexity are calculated from the data obtained for each artifact associated to a requirement. Thus, the volatility for a specific requirement is obtained from the volatility of the whole classes linked to the requirement implementation. To do this, we use the traceability matrix [1] to obtain the forward traceability information.

The 3 input variables can receive real values between 0 and 5. The output variable of the fuzzy inference system is Criticality of the Test Case (CTC). The bigger the CTC, the bigger the test case execution priority. Thus, the test case with the biggest inferred CTC is the first to be executed.

The CTC output variable can receive real values between 0 and 10, representing the real values of that variable.

To map the fuzzy sets, we used a uniform distribution of the sets. This is the normal process to do this task. After done the uniform distribution of the fuzzy sets, it is possible to improve

²http://git-scm.com/
³http://subversion.tigris.org/
We used triangular and trapezoidal membership functions to empirically map fuzzy sets to the input variables. To each input variable 3 fuzzy sets were mapped: Low (L), Medium (M) and High (H). We also used triangular membership functions to empirically map fuzzy sets to the output variable. We mapped 5 fuzzy sets to CTC: Very Low (VL), Low (L), Medium (M), High (H) and Very High (VH). We can see in details the distribution of the fuzzy sets in Figure 2.

To generate CTC of each test case through the 3 specified input variables, we must set the rule base in the proposed fuzzy inference system. The implemented rule base, which is the kernel of the CTC estimation strategy, has 27 rules. The rule base is presented in a matrix way, as seen in Table I. In this Table, R means Relevance, C means Complexity, V means Volatility, VL means very low, L means low, M means medium, H means high and VH means very high.

The rule base and the fuzzy sets associated to the input and output variables were defined using the authors’ knowledge related to software testing, but the rule base was improved with the expert knowledge as discussed in the following section. The proposed approach allows that a test engineer easily changes the test case prioritization strategy. To do that, the expert must only change the rule base of the fuzzy system.

In this work, we implemented the centroid defuzzification technique [14], formalized by Equation 1. In this equation, \( x^* \) is the defuzzified output representing the CTC value, \( \mu_i(x) \) is the aggregated membership function and \( x \) is the output variable. It is important to notice that the implemented inference system allows user to change the defuzzification method in a simplified way.

\[
x^* = \frac{\int \mu_i(x) \, dx}{\int \mu_i(x) \, dx}
\]  

V. EMPIRICAL EVALUATION

In order to evaluate the proposed approach, we have carried out an empiric evaluation. We had 2 objectives to be reached with the empiric evaluation:

1) Show the ability to mimic the expert knowledge. It means that our fuzzy inference system should prioritize the test cases of an under development system in a similar way that the expert who adjusted the rule base and the membership functions of the fuzzy inference system;

2) Show the ability to generalize the knowledge of the expert. It means that the system should prioritize different systems in a similar way.

In order to reach the above mentioned objectives, we applied our approach to prioritize test cases of 2 real information systems. The evaluated information systems are described as follows:

- **JBook**: a book loan system to control the library of a real software house.
- **ReqG**: a requirement management tool created to control the requirements and use cases of a project.

The systems were analyzed with the help of a test expert that have participated in the development of both systems. The tester is member of a software development company at Brazil. He has 2 years software development experience.

The empirical evaluation was conducted in the following order: i) the tester indicated the values for Relevance (R), Complexity (C) and Volatility (V) for each requirement related to ReqG system; ii) the tester indicated the prioritization order for ReqG tests in his point of view, using only his feeling about the features; iii) we calibrated a rule base of our fuzzy inference system with the tester information; iv) the tester indicated the values (R, C, V) for JBook system; v) the tester
indicated the prioritization order for JBook tests in his point of view, using only his feeling about the features; vi) we generated the prioritization order for JBook using the proposed approach; vii) we compared the results.

In order to calibrate the fuzzy rule base we have used linear regression [15]. This is an approach to model the relationship between a scalar variable and one or more explanatory variables. The rule base were calibrated from an estimation based on the coefficients R, C and V, suggested from the analysis of the prioritization indicated by the tester. Running a linear regression we can easily obtain the coefficients of the equation that models the behavior. The bigger the coefficient of a variable, the bigger its influence in the result. Thus, we have to reflect this in the rule base. The variable with bigger influence must be more important in the result determination of CTC.

In order to facilitate the presentation of the results, we grouped the results in two tables. Table II shows the test case prioritization inferred by our approach for ReqG system. Table III shows the test case prioritization inferred by our approach for JBook system.

The tables have the same columns. Column R shows the relevance of each requirement of the system. The column C shows the complexity of each requirement. The column V shows the volatility of each requirement. The column CTC shows the criticality of the test case which cover the respectively requirement. Finally, the column Tester shows the prioritization indicated by our expert.

The analysis of the results shows some interesting data:

- The results from the Tester and our approach, related to ReqG system, was exactly the same. This was possible due to our calibration of the rule base made by using the tester feeling. So, it was the expected result.
- The results from the Tester and our approach, related to JBook system, was almost the same. There was only two mistakes among 13 possibilities. This indicates that our approach has mimicked the Tester behavior.
- Once in both cases our proposed fuzzy inference system mimicked the Tester behavior, we also reached the other objective, that was generalize the knowledge of the expert.

The results described above validates our approach, since it is possible to capture a Tester feeling and use it for prioritization purposes. The data required to execute our approach is simple and could be obtained from automatic tools. Our approach does not requires the test execution to prioritize the test cases. This is a relevant behavior since the system source code changes all the time and it is necessary to prioritize test to be executed to assure the system behavior.

VI. RELATED WORKS

The work developed by Kavitha et al [8] presents an approach to prioritize test cases based on requirements. The approach uses three factors to prioritize test cases: the requirement priority in the client view, the implementation complexity and the volatility of these requirements. For each requirement
a factor named RFV (Requirement Factor Value) is calculated, representing the average of the considered factors. The weight of a test case is obtained through a formula that takes into account the RFV of all the requirements covered by the test case. The bigger the weight of a test case, the bigger its relevance. Our work also uses three factors to calculate the criticality of a test case, but we use a fuzzy system to mimic an expert tester.

The work of Srikanth et al. [9] proposed a technique for test case prioritization named PORT (Prioritization of Requirements for Test). PORT prioritizes system test cases based upon four factors: requirements volatility, customer priority, implementation complexity, and fault proneness of the requirements. The work presents a case study on four projects developed by students. The results show that PORT prioritization at the system level improves the rate of detection of the faults considered in the study. The method to guide the fault injection is not presented in the study. The main difference from our work is the absence of an expert evaluation in order to generate a guide for prioritization, since this could present a bias depending on the faults used for evaluation.

Yoo and Harman [4] introduced the concept of Pareto multiobjective optimization to the problem of test case selection. They describe the benefits of Pareto multiobjective optimization, and present an experimental study that investigated the effectiveness of three algorithms for Pareto multiobjective test case selection. However, the technique is not suitable for a real scenario, since it requires several test executions to find the best selection. Test cases change all the time, being hard to find a static order that is not based on properties easily obtained in an automatic way.

Maia et al. [5] proposed an alternative solution to order test cases. Such technique uses Reactive GRASP metaheuristic to prioritize test cases. They compare this metaheuristic with other search-based algorithms previously described in literature. Five programs were used in the experiments. This work has the same weakness of the previous work: the need to execute all the tests to find the best order.

As mentioned before, our work is based on a fuzzy inference system that uses common input data easily generated by automatic metric tools. The approach uses the expert knowledge to prioritize test cases and does not require the test execution to generate the suitable order. It qualifies each test case individually in order to infer its criticality. The bigger the criticality, the bigger the test case execution priority. Thus, the test case with the biggest inferred criticality must be the first to be executed.

VII. CONCLUSION AND FUTURE WORKS

This paper presents a Mamdani fuzzy inference system approach to solve regression testing prioritization problem on software projects. The approach presented differs from related works cited in Section VI because the proposed fuzzy inference system qualifies each test case individually, calculating the Criticality of the Test Case (CTC). Using the CTC, the test engineer can sort the test cases using a standard sort algorithm. In the approaches present in the related works are created a completely test case sequence. Any necessary change in the sequence requires a recalculation.

Our empirical evaluation, although simple, validates our approach, since it has shown that it is possible to capture a Tester knowledge and use it for prioritization purposes. The input data required to execute our approach is simple and could be obtained from automatic tools like SVN and Sonar. Besides, our approach does not require the test execution to prioritize the test cases, like other approaches described in the Related Work Section.

Regression testing must be executed all the time. Any approach created to solve regression testing prioritization problem must be fast and simple to be used on real scenarios. Our approach meets this requirement.

At the time that this is a work in progress, some issues in the current stage of this work do not assure the optimality of the inferred results:

- The experiment have used only one Tester. It is necessary to use a test team to generalize our results;
- It is necessary to search for alternatives to calibrate membership functions and the rule base of the proposed fuzzy inference system. We have used a linear regression in this work, but further work is needed;
- The approach must be used in a real scenario. It is necessary to create a tool able to be used by a team during a software development project.

<table>
<thead>
<tr>
<th>Requirements</th>
<th>R</th>
<th>C</th>
<th>V</th>
<th>CTC</th>
<th>Tester</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loan Request</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>6.76</td>
<td>1</td>
</tr>
<tr>
<td>Loan Report</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>6.76</td>
<td>2</td>
</tr>
<tr>
<td>Publication Register</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Loan Confirmation</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Loan Finalization</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Login</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Book Registration</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4.16</td>
<td></td>
</tr>
<tr>
<td>Loan Cancellation</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>4.16</td>
<td></td>
</tr>
<tr>
<td>Loan Cancellation Request</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3.86</td>
<td></td>
</tr>
<tr>
<td>Label Tipping Report</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3.41</td>
<td></td>
</tr>
<tr>
<td>Password Change</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2.33</td>
<td></td>
</tr>
<tr>
<td>User Registration</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2.53</td>
<td></td>
</tr>
<tr>
<td>Email Notification</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2.12</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirements</th>
<th>R</th>
<th>C</th>
<th>V</th>
<th>CTC</th>
<th>Tester</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>7.15</td>
<td>1</td>
</tr>
<tr>
<td>Review</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>7.15</td>
<td>2</td>
</tr>
<tr>
<td>Requirement</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>6.76</td>
<td>3</td>
</tr>
<tr>
<td>Use Case</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>6.76</td>
<td>4</td>
</tr>
<tr>
<td>Member</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4.48</td>
<td>5</td>
</tr>
<tr>
<td>Change History</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4.26</td>
<td></td>
</tr>
<tr>
<td>Actor</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4.26</td>
<td>7</td>
</tr>
<tr>
<td>Review Criteria</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3.50</td>
<td>8</td>
</tr>
</tbody>
</table>
Despite the issues summarized above, we can conclude, in a preliminary way, that the proposed approach is much more intuitive and suitable to be used by test engineers than other kinds of approaches based only in metaheuristics. It is possible due to the fuzzy inference system rule base construction. It mimics the test engineer knowledge simpler than other approaches heavily based on mathematical formulations.

We are developing some improvements for this work:

- a stable version of a tool that incorporates our approach and can be connected directly with SVN and Sonar tools;
- a new version of the proposed fuzzy inference system that will use Computational Intelligence to optimize the previous specified membership functions and rule base as well;
- studies to specify a metric to measure, in a formal way, the Relevance variable. The metric will be probably based on SQFD (Software Quality Function Deployment) [16].

VIII. ACKNOWLEDGMENTS

We would like to thank CNPq (grant 560.128/2010) and INES, for partially supporting this work. Also, we would like to acknowledge the Infoway company by the important contributions to the research.

REFERENCES