Experimentation on accuracy of non functional requirement prioritization approaches for different complexity projects

Raj Kumar Chopra, Varun Gupta, Durg Singh Chauhan

Uttarakhand Technical University, Dehradun, Uttarakhand, India
Amity University, Noida, Uttar Pradesh, India
GLA University, Mathura, Uttar Pradesh, India

Received 19 February 2016; accepted 6 April 2016
Available online 20 April 2016

Summary
Non functional requirements must be selected for implementation together with functional requirements to enhance the success of software projects. Three approaches exist for performing the prioritization of non functional requirements using the suitable prioritization technique. This paper performs experimentation on three different complexity versions of the industrial software project using cost-value prioritization technique employing three approaches. Experimentation is conducted to analyze the accuracy of individual approaches and the variation of accuracy with the complexity of the software project. The results indicate that selecting non functional requirements separately, but in accordance with functionality has higher accuracy amongst the other two approaches. Further, likewise other approaches, it witnesses the decrease in accuracy with increase in software complexity but the decrease is minimal.

© 2016 Published by Elsevier GmbH. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Introduction

Requirement prioritization is an activity to perform the selection of requirements, the task that is challenging due to the involvement of many stakeholders with potentially conflicting view points, multiple requirements to be handled and large effort to be invested in this activity. The wrong requirement selection not only results in wasteful effort and potentially increased effort of the next release, but also possesses the risk of project failures.

The software comprises functional and non functional requirements that together determine the acceptability of it within the market. The users never demand the non functional requirements, but appreciate if they are...
implemented. The potential reason could be that non-functional requirements determines the success of functional aspects of the system and are usually unheard amongst its users. User invests few of non functional requirements after the software is put to use and other requirements as he interacts with competitor products. For example, a mobile app with a good interface, but with slow speed will not feel appealing to the users and they may ask for fast applications. The software could have many non functional requirements that determine the success of the developed application. Resource constraints with potentially being time and cost limitation put an end to the idea of implementation of all non functional requirements and hence accuracy and effort optimized prioritization is undertaken. However, the prioritization of non functional requirements is challenging due to several reasons:

- Non functional requirements are prioritized by developers and not by users. It is important that the selection of such requirements must be aligned to the selection of functional requirements.
- Non Functional requirements are always considered as the overhead as they do not provide any functional aspect to the system. Hence, investing huge effort in their selection and implementation is considered as only overhead effort for overall development. In other words, negligible resources are allocated for non functional requirements.
- Non functional requirements shall never be prioritized with respect to functional requirements, as competitive requirements. If this happens, non functional requirements are guaranteed to get lower priority than functional requirements.
- Non functional requirements can be prioritized individually, i.e. Not in competition to functional requirements, but however their selection needs to be balanced with the selected functional requirements.

Non functional requirement prioritization approaches

The prioritization may employ existing requirement prioritization techniques using either of the two approaches:

- **Approach 1 (A1):** Prioritization of non functional requirements together with functional requirements. This option is not a good option because non functional requirements are guaranteed to lose in competition to functional aspects.

- **Approach 2 (A2):** Prioritization of non functional requirements separately from functional requirements. This approach is the good approach as mostly non functional requirements are prioritized by developers rather than users. But this is challenging because the selection of non functional requirements depends on the selection of functional requirements with which they are associated.

- **Approach 3 (A3):** Hybrid of two approaches A1 and A2. In such a scheme the non functional requirements are given separate consideration, but are selected in accordance with the prioritized functional requirements. There is no competition between non functional and functional requirements for getting implemented in the current release. Thus the selection is separate for both the two requirements, although selection depends on the functionality of the system.

**Aim and objectives of the paper**

The aim of the paper is to examine the effectiveness of the three prioritization approaches (A1, A2 and A3) for non functional requirement prioritization for different complexity project versions. To fulfill the aim, this paper is based on the two objectives, first, to examine the accuracy of the prioritization approaches by using the suitable prioritization technique on suitable software versions and second, to analyze the impact of software complexity on the accuracy of prioritization approaches.

**Experimentation**

To meet the objectives, experimentation is conducted using suitable software versions, employing a suitable prioritization technique for each prioritization approach (A1, A2 and A3). The Analytical Hierarchical Process (AHP) based cost-value prioritization technique (Karlsson and Ryan, 1997) is applied on three different complexity versions of same industrial software projects i.e. versions belonging to low, medium and high complexity. This technique is employed because pairwise comparison based prioritization technique had been found accurate by Karlsson (1996), Karlsson et al. (1998), Perini et al. (2009). The time limitation for performing the prioritization was relaxed to control the scalability variable. The scalability variable would otherwise have influenced the relation between complexity and accuracy as pairwise comparison based prioritization technique suffers from scalability issues as reported in Achimugu et al. (2014), Voola and Babu (2013), Perini et al. (2009), Ahl (2005), Karlsson et al. (1998), Karlsson et al. (2004), Lehtola and Kauppinen (2006), Ribeiro et al. (2011). The experimentation units are summarized in Table 1 as under.

The three versions of the selected project have 13 requirements (low complexity), 34 (medium) and 56 requirements (high complexity), to be subjected for prioritization.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Experimentation details.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
<td>Description of units</td>
</tr>
<tr>
<td>1.</td>
<td>Requirement prioritization technique</td>
</tr>
<tr>
<td>2.</td>
<td>Projects number and complexity</td>
</tr>
<tr>
<td>3.</td>
<td>Independent variable</td>
</tr>
<tr>
<td>4.</td>
<td>Dependent variable</td>
</tr>
<tr>
<td>5.</td>
<td>Control variable</td>
</tr>
</tbody>
</table>
Low complexity project represents first increment, medium complexity represents the fourth increment and high complexity represents an eighth increment of mass market product. The increment details i.e. the details of the number of requirements and categorization are given in Table 2. The projects are selected on the availability of project post release statistics, i.e. project success measure (sale of software increments). The selected project versions/increments were having high success rates and hence the selected requirements for the release were considered as the benchmark representing high quality prioritization. The selected project has a high number of non functional requirements implemented in comparison to functional requirements.

To measure the accuracy, the list of requirements as obtained after execution of cost-value approach on the three increments of the project individually were compared with the list of requirements implemented actually by the industry individually for different complexity projects. Large deviation between the current prioritization list and the one implemented by the industry earlier in actual for particular increment represents the less accuracy of the approach applied (A1, A2 or A3). 20 experimentation units were involved in allocation of preferences for the requirements, which are experienced software engineers with large experience in software development. Selection of requirements is a fresh process in the experimentation reported in this paper. Thus a number of requirements selected in each category and preferences need not to match those provided by industry.

### Result analysis

The execution of the prioritization technique employing three approaches individually on three complexity versions gives results that are analyzed at two levels namely:

- Comparative analysis of results for same complexity increment of three approaches gives an indication of accuracy of the approaches employed. The outcome is three results one for each complexity project. Denote the results by R1, R2 and R3.
- Comparative analysis of individual results (R1, R2 and R3) that indicates the variation of accuracy with the complexity of the software increment.

In other words, the execution of the experimentation will populate Table 3 with the results of execution as under. The results are given in Table 4.

Individual $R_i$ indicates that which approach is highly accurate for a given complexity project. Comparative analysis of $R_i$ indicates the variation of accuracy of approaches with the project complexity. The table entries are in the form of a number ($Di$). Let $N$ be the number of functional requirements and $M$ be the non functional requirements that match with the corresponding category requirement in both prioritization lists (current and those implemented by industry engineers) for a particular complexity project for a particular approach under consideration. The percentage of the number of requirements that matches is given by,
$D_i = (N + M)/T$, where $T$ is the total number of requirements for a given complexity project (Table 2), for $i = 1–9$.

Table 4 data show very interesting results. The accuracy of approach A1 decreased with an increase in complexity. The major value of accuracy is contributed by the match between functional requirements, which means non functional are ignored in competition. The accuracy of approach A2 is higher than A1 because of due consideration of non functional requirements. The accuracy decreases with the increase in the complexity of the project, which means that as the number of requirements increases, the prioritization becomes complex to execute. But still the accuracy value is high enough for high complexity projects. Approach A3 outperforms other approaches which mean that consideration should be given to non functional requirements by considering the functionality aspect. This is because non functional requirements are associated with functional requirements. The accuracy remained higher for all complexity projects. The reason is that improvement is contributed due to improvement in the selection of non functional requirements. The data show that the approach A3 outperforms other approaches and with the increase in complexity the accuracy of the approaches decreases. The decrease in accuracy is large in approach A1, lower in A2 and negligible in A3.

Conclusion and future work

Non functional requirements must be prioritized as separate entities like functional requirements, but their selection must be in accordance with the selected functionality of the software. For high complexity software the functionality based selection of non functional requirements enhances the success of the project. Selection of non functional requirements without any reference to the functionality or competitive selection has low accuracy as the complexity of the software decreases however a competitive selection has low relative performance.

In future, it is expected that the requirement prioritization techniques for the selection of non functional requirements based on system functionality will emerge in literature and real practice.

Conflict of interest

The authors declare that they have no competing interests.

Authors contributions

All the authors of this paper have equally contributed to this paper. All authors read and approved the final manuscript.

Acknowledgements

The authors are thankful to the software engineers who participated in the experiment. The authors are also thankful to the software company that shared their incremental project details with the authors to carry out experimentation.

References


