Role of Requirement Prioritization Technique to Improve the Quality of Highly-Configurable Systems

ATIF ALI 1, YASER HAFEEZ 1, SHARIQ HUSSAIN 2, AND SHUNKUN YANG 3

1 University Institute of Information Technology, Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi 46000, Pakistan
2 Department of Software Engineering, Foundation University Islamabad, Islamabad 44000, Pakistan
3 School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China

Corresponding author: Shunkun Yang (ysk@buaa.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61672080.

ABSTRACT

Highly-configurable systems are such systems which are not developed for single scenario. However, perhaps they have variable functionality and they are developed for hybrid scenarios. Producing a good highly-configurable system within time and with customer satisfaction is not easy. Handling requirements effectively in such a way that it take least time to market, is one of the most difficult tasks in highly-configurable system. In this paper, a quantitative requirement prioritization technique for highly-configurable systems is proposed. This technique involves all stakeholders and can be used primarily for large scale software projects. The proposed technique is evaluated by taking a case study of the highly-configurable point of sale for automotive industry. The result shows that the proposed technique provides promising results and can be enhanced with more future work.

INDEX TERMS

Customer value creation, highly-configurable system, quantitative approach, requirement prioritization.

I. INTRODUCTION

With the advent of technology, nowadays a lot of software systems are highly-configurable. A highly-configurable system keeps unchanged the core functionality of the software and allows the customer to customize the distinct objects of the program [1]. Therefore, it aids the customer with flexibility by providing hundreds and thousands of performance parameters. Furthermore, during the process of development; i.e., maintenance and testing, the system is customized by stakeholders such as; clients, customers, managers or designer to achieve improved functionality or greater performance [2]. Highly-configurable system (HCS) gives typical center usefulness and an arrangement of discretionary highlights to tailor variations of the options as indicated by a given arrangement of prerequisites [3]. The improvement of HCSs give clear advantages to IT businesses, encouraging the outline and execution of various software systems that offer a typical center set of capacities diminishing expenses and time-to-market to organizations. Yet, these frameworks likewise exhibit noteworthy difficulties, for example; their approval. The issue of approving a solitary design has been supplanted with considerably more difficult issue of approving the entire arrangement of designs that can be created by the greater part of the unique conceivable blends of features in HCSs [4].

HCSs are software systems which have a set of core functionalities, and some variable functionalities that are based on a set of configuration options. Changes in these aforementioned configuration options will result in a change of the program’s behavior [5].

Besides, most of the software systems are not developed for a single scenario or use case; however, they also developed for hybrid scenarios. Therefore, software systems are typically developed as configurable systems. This emergent system not only allows specific variants but also map this variant to the requirements of different application scenarios and use cases. Some elicited capabilities, named as requirements, which formalized in requirement engineering phase are the grounds for the development of a highly-configurable software system.

Though, the Requirements Engineering (RE) is the process that is necessary to develop the services or facilities that
are the needs of the customer from the system [6]. These services are inside the pre-defined constraints of functions and development. The quality product is only produced by using the high-quality requirement engineering. In the conventional approach, RE incorporates different phases, like elicitation, analysis, documentation, verification and validation, management and significantly prioritization. It tends to help in selecting the requirements that is most important to the customer at a time. This results in affecting several factors such as, price, economic constraints, accessibility of workforce, logical implementation order, client association and negative impact, etc. [4]. No other activity cripples developed system as compared to the requirement prioritization being wrong. Requirement prioritization (RP) plans system for execution in a request that attempts to increase their effectiveness to meet some performance goals, typically prioritizing the requirements as quickly as time permits. Effective RP techniques play a vital role in successful releases of software systems. Early imperfections in requirements (practical and non-useful) leave genuine operational impacts in the configuration stage. The Goal-based requirements analysis method utilizes objective topography to structure and compose such requirements data as situations, objective hindrances, and limitations.

Additionally, the quality of the HCS can only be attained by connecting the customer feedback in RP phase [4]. With this thought, advancement can be limited to more engaged prioritization consideration by the customer for one’s value creation. The furnished solution of the paper declines the gap between RP and customer value creation. Creating Customer Value expands the consumer loyalty and the client experience. Customer value helps in identifying which service or feature is important to the customer rather than focusing on the available possible option. The critical point upon where success can be declared is “customer”. Therefore, customer value creation tends to be an evolutionary process in which all diverse stakeholders declare a project as a successful project when customer satisfaction is earned [4].

Cost and time reduction to market is empowered by using the HCS architecture. Their architectures are mostly used in developing a complex software system. The software is tested for the assurance of high quality of HCSs [7]. This testing ensures that the specific feature has been tested in every possible configuration. Many test suites are required to ensure the quality assurance of HCSs. It increases the sizeable effort and time for the testing. The quality of HCS focuses on testing phase only and not in requirement prioritization which may effect on cost and resources. Therefore, the right requirement prioritization technique for the HCS is required to save resources and cost. The objectives of current research are: to extract the advanced requirement prioritization techniques, to explore the unique HCSs, to focuses on literature-based systematic mapping of extracted requirement prioritization techniques for HCSs related to software, yielded from the first part, and to propose and evaluate a novice quantitative requirement prioritization technique for HCSs.

This paper is structured as follows: section II will discuss the related work. In this section, two states of the art literature reviews are performed along with a systematic mapping. Section III emphasis on the research methodology which proposes a quantitative requirement prioritization technique. Section IV shows the results and Section V will the final section contain the conclusion and future work.

II. RELATED WORK

This section discusses the related work of the research. In the last few years, various studies have been carried out regarding highly configurable systems. Here, we present the studies which focused on implementation, testing, debugging and evolution of HCSs. We also specify how some of these studies achieved software quality in their HCSs. Additionally, we also detail the literature on requirement prioritization to enhance the credibility of our research, and to avoid the mishap of reinventing the wheel. Fig. 1 exhibits the number of studies conducted on requirement prioritization and HCSs from 1990 to 2018. To identify the gaps and gather the relevant information of aforementioned domains, we have performed a systematic literature review. Respective details of these various SLRs are discussed below.

The details of the SLR are as follows:

A. RESEARCH METHODOLOGY

The methodology of this research is four-folds: the first two parts emphasizes to explore the state-of-the-art requirement prioritization techniques and state-of-the-art HCSs, independently. The third part focuses on literature based systematic mapping of extracted requirement prioritization. The final part focuses to propose and evaluate a novice requirement prioritization technique for HCSs.

B. SYSTEMATIC LITERATURE REVIEW

The current section follows the guidelines of Barbara Ann Kitchenham to perform two different systematic literature reviews (SLR) using a protocol. The SLR explores a partic-
A. Ali et al.: Role of Requirement Prioritization Technique to Improve the Quality of Highly-Configurable Systems

TABLE 1. Systematic literature review phase.

| SLR Phases | Steps |
|------------|-------|
| Plan       | The databases are selected. |
|            | The research questions are formulated. |
|            | The inclusion and exclusion criterion is applied. |
|            | The keywords are finalized and the search strings are formulated. |
| Conduct    | Preliminary study is performed. |
|            | The data is extracted. |
|            | The data is analyzed. |
| Report     | The specifications are completed and the results are presented. |

ular area for the identification, analyzation and exploration of the information. Following is the research protocol for the identification and examination of highly configuration systems and requirement prioritization techniques, separately.

SLR is a literature-based study that considers inclusion and exclusion criteria for the exploration of particular information. It comprises of three broad phases, i.e., plan, conduct, and report. The steps under these categories are as presented in Table 1.

C. PLANNING

1) RESEARCH QUESTIONS
In order to achieve the initial objective of the study, following research questions are focused.

RQ1: What are the state-of-the-art highly-configurable systems exist in the literature?

RQ2: What are the state-of-the-art requirement prioritization techniques exist in the literature?

RQ3: What are the requirement prioritization techniques most suitable for highly-configurable systems?

2) DATA SOURCE
The search for this research was a manual and only conference and journal articles are focused, therefore, we have selected the most common and influential databases were selected based on the previous experience.

- Google scholar
- ACM
- IEEE Xplore
- Willey Inter Science
- ScienceDirect

3) INCLUSION CRITERIA
Following points are considered when selecting and finalizing the studies.

- The language of the article is considered English only.
- The publication is a must for each article.
- Only conference, journal and/or a workshop is considered.

- We selected articles related directly to our research questions.
- There should be a technique/approach/system proposed in the article.

4) EXCLUSION CRITERIA
Following points are considered when rejecting and scrutinizing the studies.

- The language of the article other than English is not considered.
- Duplication removal.
- The articles that do not discuss any technique/approach/method/system are not considered.
- Web pages, blogs, and Wikipedia articles are not the part of current research.
- The unpublished articles are also excluded.
- All those articles that discuss only the domain, however, not the technique/approach/method/system are also excluded.

5) SEARCH STRING
The search string is formulated by considering the keywords selected form research question which gets updated during the searching of relevant studies the literature. Such search strings were injected into five most common databases. Sometimes we have grouped one or more keyword by quotes, commas and parenthesis. Whenever, we changed the string the result was different. The keywords of the search strings we have used are as under:

- Highly-configurable system, customizable systems, re-configurable system, “/” OR “/limitations/” OR “/shortcomings/” OR “/practice/”.
- Requirement* prioritization technique, Requirement* prioritization categories, prioritization taxonomies, prioritization classifications, prioritization processes, prioritization method, prioritization practices.

D. CONDUCT

1) PRELIMINARY STUDY
The research papers selected during the preliminary study were refined using a tollgate approach. Following are the phases of the tollgate approach.

- Phase 1: Searching for the relevant article.
- Phase 2: Title based inclusion and exclusion.
- Phase 3: Abstract and introduction-based inclusion and exclusion.
- Phase 4: Full text-based inclusion and exclusion.
- Phase 5: Remaining articles were included in the final SLR.

2) DATA ANALYSIS
A number of requirement prioritization techniques and number of HCSs were extracted from the selected papers using the aforementioned approach. The research questions were also evaluated using the data extracted from the literature. A snapshot of article searching and selection data is shown in Table 2.
3) DATA EXTRACTION

The data extraction and related details are as follows:

a: HIGHLY-CONFIGURABLE SYSTEMS

Highly-configurable systems are software systems which have a set of core functionalities and some variable functionality that are based on a set of configuration options. Changes in these aforesaid configuration options will result in a change of program’s behavior. HCSs have been discussed across various studies in the context of implementation, testing, debugging, and evolution. Figure 2 shows the various domains of application of HCSs. Explanation for these domains is provided in the following sections.

- Automotive Software - Automotive software is related to automotive industry, i.e., making of motor vehicle. It involves both automotive products and services. System related to automotive system can be firmware, embedded software. Automotive software has a lot of configurations which are marked as feature. The requirement of the automotive software is a very important and critical as it need to handle multiple configurations. Therefore, HCS includes automotive systems.

- Embedded Software - It is a type of computer software which is used to embed in any device (e.g., watch or chip) and its main purpose is to control the device it is embedded on. The embedded software runs on special hardware and it usually have memory constrains. Most of the configurations of the embedded software are handled by the machine interface. That is why it is discussed in HCS.

- Software Product Line - Software product line concept has been around for a while. It is defined as the group of software which utilizes the common set of features which fulfills the core goal of software application. It is important to manage the requirements of software product lines so that all new requirements can be easily integrated in the existing set of requirements. Furthermore, the selection of core requirements is an important aspect of software product line systems.

- Smartphone Applications - With the advent of smartphone, the technology has change drastically. There is a variety of smartphone out there with multiple configurations such as Android and iPhone and also, the hardware is being upgraded very rapidly. the application created should support all available platform (both software and hardware) but its reality, it is difficult to support all present configuration, so requirements should be prioritized which support the maximum available configuring.

- Web Applications - Web application is a type of software application which runs on the remote server and usually a web browser is used to access the web application over the internet. With cloud computing application, a lot of configurations are now required to be supported as the cloud is offered as a foundation and Software-as-a-Service (SaaS). It is important to handle requirements for all scenarios.

- Random Software Applications - All other applications with multiple configurations are included in this category.

Table 3 specifies the details of the literature on HCSs. This table better specifies the description and limitations of the HCSs. Furthermore, Appendix A focuses the name the study of HCS and their research problem.
| S. No. | Description                                                                 | Limitations                                                                                                                                                                                                 | Ref |
|-------|-----------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| HC1   | Checked on large software’s performs regression testing only                | In case of the small test suite, the effectiveness of fault detection id decreased in the majority of minimization techniques.                                                                         | [8] |
| HC2   | LangOS implementation is a hybrid system of runtime and compile-time configurations. Initialization of all required modules is to be taken care of by a module developer. This system doesn’t have a dependency check and module bindings’ generation is limited. (Embedded OS only) | The current compile-time and a run-time implementation of LangOS are not well implemented yet.                                                                                                             | [9] |
| HC3   | This solution is constrained by the size of learning set and configuration space. It is infeasible for large systems with many configuration options. | The system can’t learn in case of a configuration option possesses complex or unsteady behavior; it can only approximate its influence. If the same program results in different performances then the system cannot reliably learn influences. | [10]|
| HC4   | The 3 basic algorithms rely on heuristics due to which the generated results aren’t always optimal. | This paper uses only one variability tool called CHEF, it should be tested on more tools for generic result.                                                                                               | [11]|
| HC5   | It only focuses on the performance aspect in limited scenarios.             | This paper provides just the overview of performance testing in the HCS. There is no implementation of the debugging system.                                                                                 | [12]|
| HC6   | It is only effective if the LSA algorithm is used for sampling. The study focuses on embedded systems only. | No debugging methodology and this paper only state the problems.                                                                                                                                             | [13]|
| HC7   | An approach used to model the statics and dynamics of systems.             | No proper empirical evaluation and results of validation are not listed.                                                                                                                                   | [14]|
| HC8   | Systems having 25 to 238K lines were used but this set is a sample of all the possible software systems. Any interactions not in the form specified in the paper are missed by the tool. | The test cases don’t have complete coverage. Individually, the test cases diverge instead of converging the multiple activities.                                                                        | [15]|
| HC9   | Only one metric for complexity was specified. Convenience sampling was used for small to medium-sized systems. | Each system was executed stand-alone. Therefore, collaborations done by the any other system would have been missed.                                                                                             | [16]|
| HC10  | The representation of variation call graph was collected from the initial analysis of only one system. | This approach wasn’t evaluated.                                                                                                                                                                             | [17]|
| HC11  | FEVER just collect meta data features of targeted objects. By working on a file-basis and parser. The foundation relies on heuristics which is not sounds. Their prototype gets shift from the Linux kernel. | The assumption-based heuristics model aids in information lost in the process. Another limitation is to validate through manual analysis.                                                                   | [18]|
| HC12  | Requires dynamic allocation of test cases depending on the feedback of the previous iteration which still results in consuming more time. (focuses only on Cyber-physical systems) | As selective items are selected, there is a high possibility that bugs are still there in the application. Also, stopping testing based on budget results in the incomplete execution of test cases. | [19]|
| HC13  | The research utilized tiny feature models to compare several sets of related features. | Only a prototype was implemented to test the algorithm and no empirical evaluation.                                                                                                                       | [20]|
| HC14  | The system’s variability deals with not only the features, functional requirements and non-functional requirements. The focus of this study is to generate an auto plant model based on the efficient development time. The solution has been validated on liquid level control systems only. | The limitation of this study is the smaller number of information in prototype.                                                                                                                                 | [21]|
| HC15  | Only available for mobile Android OS                                       | The author has not specified the type of task that has been monitored using the proposed approach.                                                                                                                                                                    | [22]|
| HC16  | Only available for Firefox on Windows, limited by the use of a single failure avoidance algorithm. | Finding workarounds quicker is difficult for this algorithm, as the act of minimizing any workaround found takes a great deal of time. There is no guarantee that the algorithm finds all of the workarounds, as some of the workarounds are more difficult to detect due to masking of configuration options. The algorithm generates false positive workarounds | [23]|
| HC17  | Only focuses on model-in-the-loop testing stage, focused on continuous controllers only, | Determination of the thoughtfulness of different factors in the model are hollow chases.                                                                                                                                                                             | [24]|
| HC18  | Only focuses on fief ox and LibreOffice                                    | There exist other sophisticated NLP and IR techniques in the literature that may produce better splitting and ranking results, which they did not explore. Some of the preference solutions proposed by the forum follow-up posts are not verifiable, and they took them as the ground truths | [25]|
| HC 19 | Learning approach lies on assumption having smooth response function with correlations in the sources and target response. | A trustworthy model can’t be learned, if the configuration factors have uneven performance conduct. [24] |
| HC 20 | Focuses on cyber-physical systems. | The solution proposed is not evaluated. [25] |
| HC 21 | A robot that allows configuration of wheels, heat sensor, humidity sensor, gas sensors, motion sensor, and sound sensors. It has 5 hours operating time and it can only go as far as 10 kms. | They claim that robot is maneuverable, but the maneuverability of robot has not been evaluated or proved in terms of degree of mobility and degree of steerability. [26] |
| HC 22 | Domain experts does not perform bug analysis; therefore, it raises a chance to identify more bugs in future mistakenly. Focuses on four HCSs (Marlin, BusyBox, Linux, and Apache). The consideration of bugs were purely dependent on the previously explored bugs, registered bugs, confirmed bugs, and bugs fixed. | The observation based on the quantity about number of bugs and their properties are missed. [27] |
| HC 23 | The system is based on the assumptions. Several configurations may vary among the declaration of functions and their bodies, therefore, errors in the code may invalidate the results. | A number of subjects used in the study for evaluation is small. [28] |
| HC 24 | Focuses on two subject systems (MixedTLS and SQLite). Features can interact on data flow level, too, but these kinds of interactions were not considered. | The representativeness of the benchmark for mbedTLS and SQLite is questionable [29] |
| HC 25 | With respect to highly configurable system, it contains several service objects having which type focused to be recognized in advance. | No evaluation of how good the suggested modeling and analysis abilities are contrasted with the conventional methodologies. [30] |
| HC 26 | Focused on Linux, BUSYBOX, and FIASCO. | The exact translation of KCONFIG is very challenging and would lead to stronger analysis, but this study uses most important logical rules only. Sometimes, the code does not maintain. UNDERTAKER aids in finding the orphaned policy comprising code with missed refractoring and alike variations in Linux. Several queries remain unprompted and one sided. [31] |
| HC 27 | Low interoperability for introducing feature models intended by “FeatureIDE”. | Obsolescence is possible (Slow to evolve) due to low interoperability with other tools. [32] |
| HC 28 | Powerful Digital Signal Processing unit required for every antenna involves specific RF units considering radio signal processing. | The resource requirements and hardware complexity are greater equaled with single antenna-based system as MIMO related HCSs cost more. [33] |
| HC 29 | Ordinary parameter standards focusing the NSGA-II algorithm were used. Several faults in Drupal v7.22 persisted in Drupalv7.23, which is considered to be bias in the fault-driven prioritization. The study is evaluated using single case study only. | Anticipated to the less testcases in the suites decreasing the performance of the system. [34] |
| HC 30 | “Configuration generation approach” deals in complete analysis, instead it provides scalable t-wise coverage. this approach works only with FM.s results are based on five SPLs. | Fault extraction and exploration is not validated and verified at any level and stage. Pro to errors generations at model’s construction stage. [35] |
| HC 31 | Agrawal et al. was referenced for the selection of mutation operators used in this research. The evaluation was not performed thoroughly. | “CheckConfigMX” bears few shortcomings. Some of the false negatives are overlooked by the system. [36] |
| HC 32 | As it uses a microscope, so calibration is required and it is light dependent. Other mechanical equipment needs to be calibrated as well. They used Windows XP and Labview8.6; in second iteration they used windows7 and Labview11. focused on plant electrophysiology. | Expensive equipment. Expensive imagery. Interferences by atmospheric conditions. Sensitivity to other ions. [37] |
| HC 33 | Only focuses on binary choices. E.g., if bugs are found then pair wise strategy will be used. Late spitting and early joining purely dependent on the internal and external dependencies of the features. Availability is limited to these systems. | In real implication showed several false positives flows. [38] |
| HC 34 | The re-engineering process is used to generate to system, with a real time case study for evaluation purpose. | Yield in multi-purpose test case prioritization dilemma in HCSs. Insufficient parameter setting might provide erroneous results. [7] |
b: REQUIREMENT PRIORITIZATION

Requirements are the features, functionalities and capabilities stated by various stakeholders that software must possess in order to consider a complete quality system. Although, requirements form the foundation of HCSs; therefore, the requirement engineering process is vital for successful development of quality software. Requirement engineering consists of various phases, namely: elicitation, analysis, documentation, verification, validation, and prioritization. The prioritization phase helps to execute the development of a system in a manner that would not only allows the timely creation of major components, but also incorporate quality and many other values into the systems. Error! Reference source not found. Table 4 lists the various types of techniques used in existing studies, whereas, Appendix B provides the name of the study that discusses the RP techniques.

Although, requirement prioritization techniques constitute a number of benefits, however; existing techniques have many limitations, including: interdependencies between requirements are not taken into account [47], [50], [53]–[58], error-prone [47]–[53], simple/informal ranking or grouping with no priority value of requirements [48], [52], [69]–[72], [61]–[68], for small number of stakeholders and could confuse stakeholders [57], [61], [64], [66]–[68], [70], [73], [74], complex [47], [54], [70], [76], [82]–[87], unscalable [55], [58], [61], [70], [76], [82], [87]–[94], time-consuming [50], [51], [94], [54], [70], [78], [84], [88], [89], [92], [93], Unsuitable for large number of requirements [50], [51], [54], [55], [58], [61], [69], [74], [76], [89], [91], [95].

After reading literature, the details of the different problems that exist in previous methodologies are as follows:

- Prioritization technique is subjective and unsystematic- The prioritization technique is qualitative and heavily relies on the expert opinion with no clear requirements on how to scale the requirements, the evaluation changes from person to person. This makes the proposed technique subjective and unsystematic.
- Prone to errors- The proposed techniques usually have a complex methodology and unclear structure. This problem leads to a lot of problems especially unreliable requirement prioritization which might lead to time delays or even the complete system failure.
- It doesn’t take interdependencies between requirements into account- If the technique does not use modular or grouping approach, then the proposed techniques must consider the dependencies between the requirements.
- It doesn’t define the value of linguistic terms to aid in the calculation of weights- In the proposed techniques, the values are not clearly defined for the scale which leads to the confusion. Either the units or selection methodology is not clearly defined or the values completely rely on the expert opinion or stakeholder choice.
- No evaluation- If the proposed techniques are not evaluated, the claims and reliability related to that technique cannot be ensured.
- Simple/informal ranking or grouping with no priority value of requirements- The techniques just utilize the expert opinion and there is no numerical assignment to the requirement or no clear ranking method to prioritize requirements.
- For small number of stakeholders and could confuse stakeholders- The techniques in which there is no

| HC | Description | Notes |
|----|-------------|-------|
| HC 35 | Focusing on pre standardization of synaptic inputs is compulsory this is due to the early overkillm and a roughly linear dependency between digital weight and PSP amplitude is required. | “PyHMF” is still overlook in gripping multiple synapses between a pair of neurons as a component of a solo projection. [39] |
| HC 36 | A web-based system focused only on knowledge-based systems. | No verifications are performed. [40] |
| HC 37 | In variability-aware testing and analyses, it doesn’t support runtime variability. “FeatureIDE” strengthens variability-aware assessment for feature-oriented software development considering Feature House | Proposed solution not evaluated. [41] |
| HC 38 | Software product lines require tests for every configuration that results massive test cases. | Results do not consider the cost of writing a reset function. [42] |
| HC 39 | It takes natural language-based questions as input. Only a single option can be selected for a single query with no tracking connections among several parameters. | “PrefFinder” is above vigorous with low rank of the resumed result. [43] |
| HC 40 | This study only discusses testing of configurable systems in general, and the challenges in testing configurable systems and how current techniques can be used to solve them. | Existing approaches for testing single configurable systems may suffer from scalability and redundancy problems. [44] |
| HC 41 | The document is made by the mixture of common system information and the configuration documents of a particular case. The invention can only document configuration parameters. | *not found* [45] |
| Sr. No | Description of Technique                                                                                                                                                                                                 | Ref  |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| RP1   | The main purpose of this research paper is to highlight the requirement prioritization in the agile method. A model is presented which prioritizes all the requirements in each iteration. The input and output of every requirement are considered a benchmark for the prioritization process. The prioritization procedure also considers clients perspective as well. | [46]|
| RP2   | This paper proposes a quantitative requirement prioritization technique which involves the client and the business organization in order to prioritize non-function requirements. It provides value to its customer by considering a particular set of the non-function requirement in the set of functional requirements. | [47]|
| RP3   | This technique uses a conventional method. It takes a requirement from all stakeholders and prioritizes them depending on user requirements and dependencies.                                                                 | [48]|
| RP4   | This technique also involves the client. It requires to communicate all the requirements before prioritizing them.                                                                                                        | [49]|
| RP5   | The main purpose of this approach is to automate the requirement prioritization process. The standard hierarchical clustering algorithm is combined with probabilistic traceability and it also incorporates the stakeholder request as well. This approach mainly focuses on finding win-win situation for all stakeholders. | [50]|
| RP6   | This approach focuses on the semi-quantitative analysis technique to estimate the priorities of the product feature. It uses machine learning and data mining approaches to prioritize the requirements. | [51]|
| RP7   | This technique involves different approaches such as AHP, self-organizing maps, cognitive psychology which accesses the knowledge of the stakeholder before the prioritization process. | [52]|
| RP8   | The requirements are sorted on the basis of the benefits they provide.                                                                                                                                                   | [53]|
| RP9   | In this approach, the requirements are prioritized on the basis of the value and the implementation cost. The pairwise comparison is performed to determine the importance of each requirement. | [54]|
| RP10  | This method scales the requirement of the stakeholder and those are converted into numerical values. A quality functional deployment technique is used to prioritize the requirement and this also represents the expectation of the clients which must be met by the developers. | [55]|
| RP11  | This method uses autonomous interactive component agents to model the requirements. The requirements are prioritized on the basis of the cost, development risk, benefit, operational risk reduction. | [56]|
| RP12  | This technique performs a pairwise comparison of the preferences. It uses machine learning to enable the prioritization process of the larger set of requirements.                                                                 | [57]|
| RP13  | This technique provides a novel way of prioritizing requirements. It uses quantitative approach to measure the requirement in all quality areas. Once all the requirements are measured, the comparison is performed to prioritize the requirements. | [58]|
| RP14  | In this approach, first all the requirements are determined from the individual’s perspective. Then it divides the requirement in both functional and nonfunctional requirements. The ambiguity and other problems are removed and, in the end, all requirements are prioritized with mutual consent. | [59]|
| RP15  | In this approach, requirements are prioritizing on the basis of the market share and the business objective.                                                                                                                                 | [60]|
| RP16  | This technique determines the importance of each requirement on the basis of the pairwise comparison matrix.                                                                                                                                 | [61]|
| RP17  | This technique also uses a value-based approach. It determines the value of each requirement from the perspective of the organization or customer and it also considers the cost of the requirement as well.                                                                 | [62]|
| RP18  | The name of the proposed method is Technique for ordering from similarity to ideal solution (TOPSIS). Requirements are valued based on a weighing scale.                                                                                   | [63]|
| RP19  | Requirements are grouped on the basis of different priorities and then numerical values are assigned to determine the priority.                                                                                                 | [64]|
| RP20  | This paper also presents an agile based requirement prioritization method. It focuses on the requirement of inter-iteration prioritization.                                                                                               | [65]|
| RP21  | It is and quantitative analytical approach which quantifies all the requirements and numerical assignment is given to the group of requirements depending on their importance.                                                            | [66]|
| RP22  | This technique also focuses on the numerical assignment of the requirements. It has 2 steps. First, it considers all the legal implications to a requirement, and then it calculates the prioritization score. It ranks the requirements from 1 to n.                              | [67]|
| RP23  | All requirements are analyzed and ranked hierarchical order (depending on dependency).                                                                                                                                     | [68]|
| RP24  | This technique involves all the stakeholders. All stakeholders are given $100 to expense on elicitation of the requirements. The requirements are prioritized by dividing total spend money with a total number of stakeholders.                                           | [69]|
| RP25  | This study uses students as the subject. They are used to prioritize requirements of a library information system and estate agency system.                                                                                           | [70]|

TABLE 4. Requirement prioritization techniques.
TABLE 4. (Continued.) Requirement prioritization techniques.

| RP   | Description                                                                                                                                                                                                 | Reference |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| RP 26 | The rationale of the requirement is used to prioritize the requirement.                                                                                                                                       | [71]      |
| RP 27 | This approach also uses curriculum-based approach for distributed stakeholders. All stakeholders are given $100 to expense on elicitation the requirements. The requirements are prioritized by dividing total spend money with total number of stakeholders. | [72]      |
| RP 28 | “Multi-Criteria Preference Analysis Requirements Negotiation (MPARN)” uses a WIN-WIN model to evaluate, negotiate and get an agreeable set of requirements by all stakeholders depending on the preferences.                  | [73]      |
| RP 29 | This paper discusses various prioritization techniques.                                                                                                                                                      | [74]      |
| RP 30 | This approach also focuses on finding win situation. If the stakeholders agree on the requirements, the win situation occurs if not, then coloration if used to prioritize the requirement matrix and find a win situation.     | [75]      |
| RP 31 | This technique provides requirement prioritization for an industrial project. It uses two different techniques. First, it evaluates the requirements using the pairwise comparison. Then it uses the Wiegers’ method to handle change requests. | [76]      |
| RP 32 | This technique also involves stakeholders. If the one stakeholder requirement is same for all, it is considered the win situation; else iteration is made to find a win situation.                                | [77]      |
| RP 33 | The requirements are iteratively prioritized using the automated fuzzy logic-based system. It involves stakeholders and experts as well.                                                                     | [78]      |
| RP 34 | It uses Analytic Hierarchy Process for the prioritization of the requirements. It exploits the advantages of machine learning to prioritize large set of requirements. Also, it uses pairwise comparison and Boolean matrix for accurate results. | [79]      |
| RP 35 | SWTMetrics is a tool that works for requirement change management.                                                                                                                                          | [80]      |
| RP 36 | The requirements are changed into the fuzzy multi-attribute decision problem. Once it is done, it uses the expected value operator is used to ranking the alternatives listed in the problem formulation.             | [81]      |
| RP 37 | This approach uses a tree-based weighting scale approach. Each node is searched for a requirement which is compared to the other requirement on the basis of the scale.                                          | [82]      |
| RP 38 | Contribution values and preference matrices are used for each requirement. A goal-based graph is produced to determine the high priority requirement.                                                               | [83]      |
| RP 39 | This technique also uses the Analytic hierarchy process: A pair-wise comparison matrix to calculate the relative importance of each requirement.                                                                     | [84]      |
| RP 40 | In this approach, each stakeholder chooses top 10 requirements.                                                                                                                                                | [85]      |
| RP 41 | An applying quality checks is a technique used in the process of supported negotiation groupware and in other techniques of an inspection of results of negotiation.                                                | [86]      |
| RP 42 | The technique of experts and the fuzzy logics-based system that are automated involve stakeholder for prioritize requirements iterative. The technique is based on an algorithm which is used to search the requirements at nodes and to compares them to find out the required importance by using scale of weight. | [87]      |
| RP 43 | Proposed simulation based on empirically studies of agile and plan based requirements of the studies of management. The simulation stated that of many claims that the agile methodology works best when the requirements have flexibility but in plan based method works good in case if requirements are constant and it is best to combine the strategies that produce best for a typical effort of development, to Prioritize the requirements on the bases cost of implementation and perceived value. | [88]      |
| RP 44 | Pair comparison technique of matrix is used to measure the necessary requirement.                                                                                                                              | [89]      |
| RP 45 | The technique of case-based ranking uses approaches of machine learning which reduced all the required information of stakeholders to achieve the given quality degree ranking.                                   | [90]      |
| RP 46 | The requirement uncertainty prioritization approach (RUPA) is a technique which based on the priority of requirements to aggregate its weight by using an algorithm of an internal evidential reasoning (IER).                                               | [91]      |
| RP 47 | The technique formats the group priority in which requirements are classified on the bases of stakeholder’s importance.                                                                                           | [92]      |
| RP 48 | Pair comparison technique of matrix is used to measure the necessary requirement.                                                                                                                              | [93]      |
| RP 49 | Pair comparison technique of matrix is used to measure the necessary requirement.                                                                                                                              | [94]      |
| RP 50 | To utilize the method of matrices chronologically represent the expectations of client and that met the expectations of developers.                                                                            | [95]      |

significant involvement of the stakeholder. If multiple people are responsible for many roles, this might lead to confusion where all distinct stakeholders are required to make certain decision.
Conceptual model- The conceptual model is missing or unclear for the proposed technique. This problem persists if the proposed technique is not similar to a proposed technique.

Semi-quantitative- The techniques, which are not quantitative or semi quantitative leads to the ambiguous requirement prioritization which might lead to error.

Difficulty in reaching consensus- The negotiation-based techniques require all clients to sit and communicate with one another. In these techniques, there is a chance that stakeholders might not reach at some conclusion.

It doesn’t consider user need- The client is not involved in the process which leads to the customer dissatisfaction. To create customer value, client must be involved in the prioritization phase.

Negative correlation not managed- Increase in one factor might lead to decrease of some other important factor. These types of relationships are not handled in the proposed techniques.

Aggregation strategy of attributes don’t capture conceptual properties of membership and similarity- All stakeholders are not involved or wither the attributes which are considered important for the requirement prioritization, are not used in the proposed methodology.

Complex- Complex prioritization techniques which cannot be utilized without special training or workshop.

Unscalable- The proposed methodology has certain constraints which prevents its utilization for the large-scale projects.

Time consuming- The requirement prioritization technique takes significant time from software development life cycle as it is a complex and lengthy process or there is no automation support for the proposed technique.

Stakeholders can manipulate results according to their own objective- When the whole prioritization process depends on the stakeholders, there is a threat that results might be manipulated by the stakeholder for their own motives.

Unsuitable for large number of requirements- The proposed methodology is not scalable and can be utilized only for the limited size projects or projects of specific domain. Table 5 below gives a mapping of limitations in existing RP techniques.

The variable nature of HCSs can put forth various challenges. One of these challenges is to fulfill all not only major requirements but also to gain customer satisfaction. This is hard to achieve considering many unique arrangements in which number of features can be placed and used. We can leverage the strengths of requirement engineering to better handle this scenario. Requirement Engineering do not only help us in identifying the most important requirements as specified by customer, but it also ensures on-time delivery, decrease the system’s cost, and improve chances of project’s success. Nevertheless, the development of a quality system is highly dependent on the amount of customer input at the requirement gathering and prioritization phase. Although, the requirement prioritization and customer value creation are two separate paradigms, however, they can significantly affect one another. Therefore, we want to propose a method of delivering a good quality HCS by incorporating maximum customer feedback during the requirement prioritization phase.

III. RESEARCH METHODOLOGY

A. LEAST SQUARES ESTIMATION

The least squares abbreviated as (LS) is a technique focuses the minimization of squared differences between empirical data and their desired values for assessing preferred parameters. Considering the regression analysis, it comprises a response variable usually represented by ‘Y’ and one or multi-covariables usually represented by ‘X’. The response variable purely dependent on the variation in covariables. For example, change in grades ‘Y’ are mostly due to the change in students’ skills and persistence ‘X’. Similarly, variant of life ‘Y’ are predominantly up for the modifications in eco-friendly settings X. Therefore, for all values of covariables ‘X’, the finest projection of Y considering mean-square-error represents mean \( f(X) \) studying provided X. We can consider this as:

\[
Y = f(X) + \text{noise}
\]

\( Y \) is a function of X including noise, whereas, \( f(X) \) is a regression function, computed from selecting n multi-covariables and corresponding response-variables \( (x_1, y_1), \ldots, (x_n, y_n) \).

Let’s consider \( f \) is a finite number \( p \leq n \) having parameters \( \beta = (\beta_1, \ldots, \beta_p) \), i.e., \( f = f_\beta \). The \( \beta \) is computed from least-squares predictor \( \hat{\beta} \) providing data super-fit by diminishing entire feasible \( b \).

\[
\sum_{i=1}^{n} (y_i - f_b(x_i))^2
\]  

(1)

Let’s suppose, we have a scenario in which \( f_\beta \) is a linear function of \( \beta \), i.e.,

\[
f_\beta(X) = X_1\beta_1 + \cdots + X_p\beta_p
\]  

(2)

where, \( X_1 + \cdots + X_p \) are the multi-covariables of \( f_\beta(X) \).

We shall follow matrix notation for (2). So, we have \( y = (y_1, \ldots, y_n) \). Similarly, \( X \) represents \( n \times p \) data matrix. Whereas, \( n \) are the observations for variables \( p \).

\[
X = \begin{pmatrix}
x_{1,1} & \cdots & x_{1,p} \\
\vdots & \ddots & \vdots \\
x_{n,1} & \cdots & x_{n,p}
\end{pmatrix} = (x_1 \cdots x_p)
\]  

(3)

where, the \( X_j \) shows the column having \( n \) observations on variable \( j, j = 1, \ldots, n \).

\( v \) is an n-dimensional vector and \( \|v\|^2 \) is the length power two of \( v \) which is further represented by \( \|v\|^2 = v'v = \sum_{i=1}^{n} v_i^2 \). Therefore, (1) could be like:

\[
\|y - Xb\|^2
\]
TABLE 5. Limitations of requirement prioritization.

| No. | Problems                                                                 | Reference            |
|-----|---------------------------------------------------------------------------|----------------------|
| 1   | Prioritization technique is subjective and unsystematic                    | [46]–[48]            |
| 2   | Prone to errors                                                           | [49]–[53]            |
| 3   | It doesn’t take interdependencies between requirements into account        | [47], [50], [53]–[58]|
| 4   | It doesn’t define the value of linguistic terms to aid in the calculation of weights | [59], [60]         |
| 5   | No evaluation                                                             | [61]                 |
| 6   | Simple/informal ranking or grouping with no priority value of requirements | [48], [52], [61]–[72]|
| 7   | For small number of stakeholders and could confuse stakeholders            | [57], [61], [64], [66]–[68], [70], [73], [74] |
| 8   | Conceptual model                                                          | [65]                 |
| 9   | Semi-quantitative                                                         | [66]                 |
| 10  | Difficulty in reaching consensus                                           | [75]–[79]            |
| 11  | It doesn’t consider user need                                              | [80]                 |
| 12  | Negative correlation not managed                                          | [75]                 |
| 13  | Aggregation strategy of attributes don’t capture conceptual properties of membership and similarity | [81]             |
| 14  | Complex                                                                   | [47], [54], [70], [76], [82]–[87] |
| 15  | Unscalable                                                                | [55], [58], [61], [70], [76], [82], [87]–[94] |
| 16  | Time consuming                                                            | [50], [51], [54], [70], [78], [84], [88], [89], [92]–[94] |
| 17  | Stakeholders can manipulate results according to their own objective      | [51], [84]           |
| 18  | Unsuitable for large number of requirements                                | [50], [51], [54], [55], [58], [61], [69], [74], [76], [89], [91], [95] |

FIGURE 3. The projection of the vector \( \mathbf{y} \) on the plane spanned by \( \mathbf{X} \).

It shows the distance between the both columns \( \mathbf{Y} \) and \( \mathbf{X} \) and joining the \( \mathbf{b} \) which is the linear combination. Figure 3 shows the distance that could be lessened considering the projection overhang of \( \mathbf{y} \) on \( \mathbf{X} \). By taking the inverse we have Eq. 4 showing the variance:

\[
\hat{\mathbf{\beta}} = (\mathbf{XX})^{-1}\mathbf{Xy}
\]  

Then, we have provided \( \mathbf{X} \), the value of \( \hat{\mathbf{\beta}} \) is equivalent to

\[
(\mathbf{XX})^{-1}\sigma^2
\]

Here, \( \sigma^2 \) is noise variance. Considering the estimator of \( \sigma^2 \), the equation could be like:

\[
\hat{\sigma}^2 = \frac{1}{n-p} \left\| \mathbf{y} - \mathbf{X} \hat{\mathbf{\beta}} \right\|^2 = \frac{1}{n-p} \sum_{i=1}^{n} \hat{e}_i^2
\]  

Here, \( \hat{e}_i \) shows some residuals.

\[
\hat{e}_i = y_i - x_{i,1}\hat{\beta}_1 - \cdots - x_{i,p}\hat{\beta}_p
\]  

E.g., estimating the variance of \( \hat{\beta}_j \) will be

\[
\text{var}(\hat{\beta}_j) = \tau_j^2 \hat{\sigma}^2
\]

where \( \tau_j^2 \) represents a particular element diagonal to \( (\mathbf{XX})^{-1} \). Taking least squares estimator to approach Equation (7):

\[
\hat{\beta}_j \pm c\sqrt{\text{var}(\hat{\beta}_j)}
\]  

Whereas, \( C \) is the constant highly dependent on particular confidence level (\( \alpha \)). Having value of \( \alpha \) 95%, we have \( c = 1.96 \) considering a decent estimation while the value of \( n \) is larger. On contrary, a moderate \( c \) from student distribution tables can be considered \( n - p \) degrees of freedom.

Keeping in mind the limitations and problems identified in Table 3 and Table 5, we are proposing a requirement prioritization technique which will help to produce a good quality HCS. It will result in creating customer value creation. For this, we are proposing a quantitative requirement prioritization technique for the highly configuration system which
A. Ali et al.: Role of Requirement Prioritization Technique to Improve the Quality of Highly-Configurable Systems

FIGURE 4. Proposed methodology.

uses combination of curriculum-based and ranking method to determine the high priority requirements. This technique considers all stakeholders, even distributed stakeholders, in order to determine the high priority requirement. For this proposed approach, we need to define three Matrixes; named as, Factor Priority Matrix (Matrix X), Number Assignment Matrix (Matrix Y) and Assessment Matrix (Matrix Z). The details of the proposed methodology are defined as follows:

- Firstly, all requirements of the HCS are collected from the clients.
- Feasibility analysis is performed whether the given requirement is in given budget and time constraint or not. These resources and scared and any wrong estimation may result in loss or even the system failure. Once the feasibility of the project is performed, convert the requirements to IEEE Software requirement specification format and then we proceed to the next step.
- After the feasibility analysis, the factor priority matrix is used to separate the requirements into different categories and provide a scale to some factors for each requirement. These factors and method of scaling are discussed in detail, later in this section.
- After that, value assignment matrix is considered for the determination of the numerical prioritization estimate of the provided requirement.
- Give evaluation matrix form to all experts to determine the mean result and prioritize result on the basis of collective information of all the team members.
- If the prioritized requirements have a tie, use curriculum-based method to break the tie by involving all stakeholders.

The proposed methodology is show in Figure 4, and it is followed by the detailed procedure of all the matrixes.

1) MATRIX X: FACTOR PRIORITY MATRIX
This Matrix is comprised of two different approaches, i.e., setting the prioritized feature list for the requirements [92] and sample quantification matrix to prioritize the N requirements [92]. This proposed approach conforms to the IEEE Software Requirement Specification standard. This approach uses the structured matrix approach to prioritize the requirements for HCSs which comfort to the standards. The requirements are converted to IEEE SRS standard format after the collection process.

Afterward the conversion process, the requirements are separated into functional requirements, system requirements, constraints and system Layout. Functional requirements include all the modular requirements for the system. Whereas, the system requirements are comprised of non-functional requirements; such as, performance, scalability
and reliability, etc. Limitations involve different constraints while layout contains all the interfaces, for example, hardware interface, software interface, network interface, etc. All the requirements are assigned a unique index which is used as the identification number. All those requirements which are given the unique identification number are considered for the prioritization. In the next step, the requirements are evaluated for different factors which are; Time, Cost, Complexity, Criticality and Dependency. These factors are identified on the basis of triple constraints; time and cost and scope, whereas, the quality is considered by other three factors which are complexity, criticality and dependency. These factors are measure on the 3-point Likert scale with the values (level 1) high, Level 2 (medium) and Level 3 (low). Once the requirements are evaluated for the aforementioned factors, the precedence is assigned to the requirements. The precedence is set as the critical, urgent, important, normal and optional. The structure of the Matrix X is as shown in Table 6.

2) MATRIX Y: NUMBER ASSIGNMENT MATRIX
This matrix is used to provide all the possible values combination for the levels provided in the Matrix A. The statistical ranking method is used to rank the items. In order to achieve this, reverse statistical ranking method is used to prioritize the requirements. A table is generated for the possible values by using the formula $X^n$, with numerical value 35, as there are total of $x = 5$ factors for total combinations are 243. The sample of Matrix Y is shown in Table 7.

To fill up this table is a troublesome task; therefore, a machine learning algorithm is used to generate a tree for the possible combinations and values. The pseudo code is shown above.

Until now, we have assigned the values to the given requirements by using matrix Y. Now, in the matrix X, the value column will be filled by the different team members involved in the requirement process to give a value to each of the factors identified in Matrix Y. A sample of matrix Z is as shown in Table 8.

3) MATRIX Z: ASSESSMENT MATRIX
The procedure to fill up the matrix Z is as follows:

Step 1: Once the requirement in the matrix X are assigned the identification number, the matrix Z is provided to all the team members.

| Requirements | Factors | Value from Matrix Y (As per Nature) |
|--------------|---------|-----------------------------------|
| Type of Requirement | Component Index | Time | Cost | Complexity | Criticality | Dependency |
| Functional Requirements | Component A | R1 | Level-1 | Level-1 | Level-1 | Level-1 | Level-2 | 728 |
| Functional Requirements | Component B | R2 | - | - | - | - | - | - |
| System Requirements | Environment Variables | R3 | - | - | - | - | - | - |
| System Requirements | Platform | R4 | - | - | - | - | - | - |
| Limitations | Software Constraints | R5 | - | - | - | - | - | - |
| Limitations | Hardware Constraints | R6 | - | - | - | - | - | - |
| Layout | User Interface | R7 | - | - | - | - | - | - |
| Layout | Network Interface | R8 | - | - | - | - | - | - |

| Time | Cost | Complexity | Criticality | Dependency | Nature |
|------|------|------------|-------------|------------|--------|
| Level-1 | Level-1 | Level-1 | Level-1 | Level-1 | Trivial: 243, Minor: 486, Major: 729, Urgent: 972, Showstopper: 1215 |
| Level-1 | Level-1 | Level-1 | Level-1 | Level-2 | Trivial: 242, Minor: 485, Major: 728, Urgent: 971, Showstopper: 1214 |
| Level-1 | Level-1 | Level-1 | Level-1 | Level-3 | Trivial: 241, Minor: 484, Major: 727, Urgent: 970, Showstopper: 1213 |
| Level-1 | Level-1 | Level-1 | Level-2 | Level-1 | Trivial: 240, Minor: 483, Major: 726, Urgent: 969, Showstopper: 1212 |
| Level-3 | Level-3 | Level-3 | Level-2 | Level-3 | Trivial: 4, Minor: 247, Major: 490, Urgent: 733, Showstopper: 976 |
| Level-3 | Level-3 | Level-3 | Level-3 | Level-1 | Trivial: 3, Minor: 246, Major: 489, Urgent: 732, Showstopper: 975 |
| Level-3 | Level-3 | Level-3 | Level-3 | Level-2 | Trivial: 2, Minor: 245, Major: 488, Urgent: 731, Showstopper: 974 |
| Level-3 | Level-3 | Level-3 | Level-3 | Level-3 | Trivial: 1, Minor: 244, Major: 487, Urgent: 730, Showstopper: 973 |

| Type of Requirement | Component Index | Time | Cost | Complexity | Criticality | Dependency |
|---------------------|-----------------|------|------|------------|-------------|------------|
| Functional Requirements | Component A | R1 | Level-1 | Level-1 | Level-1 | Level-1 | Level-2 |
| Functional Requirements | Component B | R2 | - | - | - | - | - |
| System Requirements | Environment Variables | R3 | - | - | - | - | - |
| System Requirements | Platform | R4 | - | - | - | - | - |
| Limitations | Software Constraints | R5 | - | - | - | - | - |
| Limitations | Hardware Constraints | R6 | - | - | - | - | - |
| Layout | User Interface | R7 | - | - | - | - | - |
| Layout | Network Interface | R8 | - | - | - | - | - |
TABLE 8. Matrix Z – assessment method.

| Type of Requirement | Features/modules | Req. Index | Value column from matrix A of each member | Mean |
|---------------------|------------------|------------|------------------------------------------|------|
| FRs                 | Module A         | A1...      | -                                        | -    |
|                     | Module B *All Modules | B1...      | -                                        | -    |
| NFRs                | Usability        | C1C2 ...   | -                                        | -    |
|                     | Performance *All NFRs | D1D2 ...   | -                                        | -    |
| Design Constraints  | Software constraints | E1 E2 ...  | -                                        | -    |
|                     | Hardware constraints | F1F2 ....  | -                                        | -    |
| Interfaces          | User Interface   | G1 G2      | -                                        | -    |
|                     | Hardware Interface | H1 H2      | -                                        | -    |

FIGURE 5. Curriculum based method to resolve conflicting priorities obtained from Matrix Z.

Step 2. All the team members are responsible to provide a value to each factor in the requirement of Matrix X. They assign a criterion to every requirement such as trivial, minor, major, and urgent or showstopper.

Step 3. Once the nature of requirement is identified, they assign a value to it using Matrix Y.

Step 4. As soon as all the values are assigned, find the mean value of the requirement mentioned in the matrix Z.

Step 5. Sort the final value column from highest to lowest value. The formula to find the mean value for each requirement is as follows:

\[
\text{Requirement Priority Value} = \frac{\sum_{k=0}^{n} (\text{Stakeholder value for the requirement})}{\text{Total Number of Stakeholders}}
\]

Step 6. Plot a graph for presentation.

If there is a tie in the mean values of the requirements, the conflict is broken using the curriculum-based approach. Each stakeholder is given fabricated $100 and then in the tied top prioritized requirements, the stakeholder set a price for the requirement from that money. The top requirement is calculated by the following equation.

\[
\text{Average Cost of Requirement} = \frac{\sum_{k=0}^{n} (\text{Cost Assigned by each stakeholder})}{\text{Total Number of Stakeholders}}
\]

The requirement with the highest average cost is given the highest priority. If the conflict remains, the process is repeated again unless the tie is broken. The procedure to resolve conflict is shown in Figure 5.

The main advantages of this technique are as follows:
- This technique is not subjective i.e., it is not limited to just HCS; however, it can be used for any. Moreover, it can work for project of any size and any domain which makes it scalable.
- The methodology clearly defines each step hence, it is less prone to error.
- The linguistics for giving numerical values to the requirements is clearly defined.
- It does not influence by the dependencies between the requirements; therefore, it easily prioritizes requirements without any hassle.
- It is easy to reach a consensus due to the hybrid approach which forces the stakeholders to make an unbiased decisions.
- It considers all the user needs and involves all stakeholders.
- This technique can be easily automated to decrease the prioritization time.

B. ESTIMATION USING LSE

In our work, we take \( Y_N \) as values for \( p \) number of factors affecting a requirement and \( N \) represents number of requirements. The value of any factor can be of any order depending upon its priority which is given by \( q_n \).

\[
y_N = \sum_{n=0}^{p} \frac{q_n}{y_n} \text{ where } 1 \leq q_n \leq p
\]
Function HCS_RP_TREE Arr [243]; array containing all possible value
MI: MI Index of Node
TV: Top_value
BV: Bottom_Value
CV: Current_value
LN: Leaf_Node
TIV: Temporary_INT_Value

return Value_Tree

//Conditions
Begin:
If CV equals "Trivial" assign $0$ to CV and return CV;
If CV equals "Minor" assign 243 to CV and return CV;
If CV equals "Major" assign 486 to CV and return CV;
If CV equals "Urgent" assign 729 to CV and return CV;
If CV equals "ShowStopper" assign 972 to CV and return CV;
End

;This loop is repeated 5 times, once for each fato cost, time, complexity, criticality and dependency

Loop for 5 iterations start from 1
TIV is assigned a level value i.e. Level 1, Level 2 or Level 3
If TIV is equal to "Level 1"
  MI = MI / 3
 BV = TV - MI + 1
End
If TIV is equal to "Level 1"
  MI = MI / 3
  Bottom_Value(BV) = Top_Value(TV) - MI + 1;
End
If TIV is equal to "Level 2"
  MI = MI / 3
  Minimum = Bottom_Value(BV);
  Top_Value(TV) = Bottom_Value(BV) + MI; Bottom_Value(BV) = Top_Value(TV) - MI;
End
If TIV is equal to "Level 3"
  MI = MI / 3
  Top_Value(TV) = Bottom_Value(BV) + MI - 1;
End
If last iteration of loop is true
  If Top_Value is equal to Bottom_Value
    Assign value of array Arr at index [Top_Value] To Leaf_Node (LN);
End
End
Return Leaf_Node + Current_Value;

// specifies priority of the requirement
End HCS_RP_TREE;

Now, we multiply the values of each requirement with the values given by the experts. We want to estimate the requirement priority values $\hat{V}_{rM}$ after getting our first idea from the experts as represented in the equation below where M represents the $M^{th}$ expert.

$$\hat{y} = (XX)^{-1}XV_{rM}$$

$$\hat{V}_{rM} = X\hat{y}$$

This will not only provide us the estimate but also, we will be able to learn about an individual expert. Once we get the requirement priority value of one requirement, we can get the other values similarly making the matrix of all M requirements and N experts.

$$\text{Requirement Priority Value} = \frac{1}{N} \sum_{M=1}^{N} V_{rM}$$

Suppose, we have an equation of multi-regression comprised of multiple variables and constants as:

$$V_{rM} = \beta_1 + \beta_2X + \beta_3X^2$$
Let we have $M = 100, \epsilon_i = i/M$ whereas, $i$ shows the values from $1$ to $N$. By reshaping Matrix we have

$$X = \begin{bmatrix} 1 & \epsilon_1 & \epsilon_1^2 \\ \vdots & \vdots & \vdots \\ 1 & \epsilon_M & \epsilon_M^2 \end{bmatrix} = \begin{bmatrix} 1 & 1/100 & 1/(100)^2 \\ \vdots & \vdots & \vdots \\ 1 & 1 & 1/(100) \end{bmatrix}$$

This gives

$$X'X = \begin{bmatrix} 100 & 50.5 & 33.8350 \\ 50.5 & 25.5025 & 20.5033 \\ 33.8350 & 25.5025 & 20.5033 \end{bmatrix}$$

$$(X'X)^{-1} = \begin{bmatrix} 0.0937 & -0.3729 & 0.3092 \\ -0.3729 & 1.9571 & -1.8189 \\ 0.3092 & -1.8189 & 1.8009 \end{bmatrix}$$

After the simulation of $M$ independent variables $e_1, ..., e_M$, computed for $i$ range from $1$ to $M$:

$$V_i = 1 - 3x_i + e_i$$

Therefore, considering same example the parameters will be like:

$$y = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -3 \\ 0 \end{bmatrix}$$

Moreover, $\sigma^2 = 1$, as for simulation purpose the values are not unknown. We computed $X'y$ as the least square estimator. So,

$$X'y = \begin{bmatrix} -64.2007 \\ -52.6743 \\ -42.2025 \end{bmatrix}$$

the value of $\hat{y}$ will be

$$\hat{y} = \begin{bmatrix} 0.5778 \\ -2.3856 \\ -0.0446 \end{bmatrix}$$

The value of $\hat{\sigma}^2$ was also computed which was 0.883.

$$\hat{\sigma}_{VM} = \begin{bmatrix} 1 & 1/100 & 1/(100)^2 \\ \vdots & \vdots & \vdots \\ 1 & 1 & 1/(100) \end{bmatrix} \begin{bmatrix} 0.5778 \\ -2.3856 \\ -0.0446 \end{bmatrix}$$

$$\text{Requirement Priority Value} = -0.642$$

### IV. RESULTS AND DISCUSSIONS

The current approach is not focusing on the comparison with the prior techniques, this is because none of the requirement prioritization techniques are compatible with highly configurable system as we have extracted limitations in the former requirement prioritization techniques mentioned in the Table 5. However, to evaluate the proposed methodology, two different processes are followed. Firstly, a case study of highly-configurable point of sale system was considered. Secondly, focus group was considered to score on several parameters which were extracted in the aforementioned section and presented in Table 3.

With the perspective of case study, the system was capable of being customized for all automobiles industries. All the requirements were already gathered from the clients. The requirements were divided into 10 distinct modules. Further, the IEEE standardized SRS document was created for requirement specifications. The considerable system requirements were privacy, scalability, usability, reliability and maintainability. There were some software constraints involve there as limitations. The main layout that was handled was user interface.

For the evaluation perspectives, a group of 5 domain experts were selected. This included a team lead, a requirement analyst, 2 developers and a tester from a private software company. The main task for the participants was to prioritize the given requirements by following the procedure. A two-hour workshop was given to make them completely understand the prioritization process. First, matrix $X$ was formed by all the participants. The values were gathered from the matrix $Y$. The results concluded from matrix $Z$ are represented in the Table 9.

For all the results of the participants, mean value is computed against each requirement. After this, a plot is generated to prioritize and represent requirements. Prioritized requirements are shown in Figure 6. As, we can clearly see that A3 requirement has the highest priority followed by the I1 requirement. In system requirements, privacy is more important than any other attribute. Therefore, the requirements are successfully prioritized quantifiably.

With the perspective of Focus group, a questionnaire was designed with items extracted from the mapping of requirement prioritization techniques. The items are as shown in Table 5. These items are the existing problems found in the literature of requirement prioritization Techniques. Therefore, it was mandatory to cross check our proposed technique against such problems. In order to achieve this task, a focus group comprising eight members with their experience of 3 to 11 years in the industry was considered. All the members had a questionnaire consisting of all the items need to be score against the Likert scale before the sitting. One of the authors of the research had discussed the working of the proposed technique when the session was actually started. After this presentation, members of the focus group were requested to fill up the questionnaire and ask the queries if there was a need. This session took a total of 45 minutes. The result of the focus group is shown in Figure 7.

In order to analyze the result, descriptive statistics is applied by using SPSS 20.0. The result of focus group is shown in the Table 10. According to the result, all the members of the focus group were agreed on the proposed technique that it is prone to error, support modifiability and suitable for large scaled projects. Moreover, they were also agreed on that it is hard to reach on the consensus. Therefore, curriculum-based approach was used to solve this problem. Similarly, the members were also agreed on that proposed technique is focusing on requirement ranking. Furthermore,
focus group members were not agreed on the proposed technique that it takes requirement interdependencies into the account. They were undecided about the value creation of the customers. They were somewhat dominating towards agree about the complexity, time consuming and scalability of the technique.
This technique covers a lot of problems from the existing techniques. Following are some strengths of the proposed technique:

- No matter the size of requirements, this proposed technique can prioritize any number of requirements regardless of the dependencies.
- There is no repetition in the ranking and different ranking is assigned to each requirement.
- Focus on the five main factors which impact any requirement.
- Following and SRS standard will automatically create a conformity standard for all the stakeholders.
- Every stakeholder can be involved in the process.
- Clearly defined steps with no room for ambiguity.
- Ensure a good quality HCS is produced.

V. CONCLUSION AND FUTURE WORK

Highly-configurable systems have changed the course of the technology. It allows customers to customize the distinct objects of program according to their desires. Producing quality highly-configurable software is a difficult task. A good requirement prioritization technique can help produce a good HCS which can satisfy all customer requirement in time and within budgets.

In this research paper, we have proposed a quantitative requirement prioritization technique for the HCS. It involves all the stakeholders and can work for any number of requirements. The stakeholder team first assign levels to the requirements, after that, numerical values are assigned to the requirement. All stakeholder gives values to the requirement and then mean of those value against each requirement.
are plotted in the graph to get the highly prioritize requirement.

The leading advantage of the proposed methodology so the conformance of time saving, producing a quality HCS, prioritization of requirement regarding the total number of requirements. All basic scenarios have been covered in this technique. Some complexities are left. We shall try to focus them in our future work. This research can be applicable to prioritize requirements in the domain of internet of things. Additionally, it is also suitable for scaled software development environment.

This study comprises some potential threats to validity, i.e. the case study is just one project. The change in project may change the results. Further, there could be the biasness of the stakeholders voting being human. Change in the type of HCS or change in the evaluation process may produce different results.

We will incorporate the change of requirements in this technique and how the requirement changes will be handled using this approach is a part of our future work. Moreover, we will fully automate the requirement prioritization process by creating a tool and then evaluate against the state-of-the-art existing solutions. Additionally, it will be focused to apply same methodology for IoT based and scaled software requirements prioritizations. Finally, we will also incorporate machine learning algorithms such as regression and classification which will improve the accuracy of the process.

**APPENDICES**

**APPENDIX A**

| S. No | Name                                                                 | Research Problem                                                                                                                                 |
|-------|----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| HC1   | "Detecting and Reducing Redundancy in Software Testing for HCSs"    | One of the fault tolerance methods is identifying the redundancy in software development but it results in decreased testing performance and increased effort |
| HC2   | "LangOS - A Low Power Application - Specific Configurable Operating System" | An application for wireless sensor node when implemented has many factors affecting it, i.e., selected platform, limited resources of microcontroller etc. Although there are many requirements a developer prefers to use a single operating system but a full-fledged OS cannot be implemented on a sensor node due to limited resources. |
| HC3   | "Performance-Influence Models for HCSs"                              | Configurability is beneficial but it forwards various challenges in terms of performance, testing and debugging. It is difficult to test a configuration option individually. And testing various combinations of configuration options can give rise to testing space of exponential size. |
| HC4   | "Presence-Condition Simplification in HCSs"                          | This system analyses only valid combinations of configuration options and avoids the present condition.                                          |
| HC5   | "An empirical study on performance bugs for a HCS"                  | In HCS, testing performance is difficult because of the large configuration space and hence many possible inputs and execution environments.       |
| HC6   | "Safely evolving pre-processor based configurable system"           | Due to lack of understanding of sampling algorithms, developers introduce bugs during the evolution of pre-processor based HCSs. This decreases code quality and increases the cost by increasing the time taken to remove those bugs. There is a need to safely evolve pre-processor-based configurable systems. |
| HC7   | "Limiting recertification in HCSs"                                  | Due to the vastness of the configuration space, the recertification of every configuration is difficult.                                     |
| HC8   | "Gen: Dynamic Interaction Inference for Configurable Software"       | Developers need a deep knowledge of HCSs’ configuration space so that they could develop, evolve and analyze the system. But acquiring this knowledge is time-consuming and requires significant effort. |
| HC9   | "On Essential Configuration complexity: Measuring Interactions in Highly-Configurable Systems" | To avoid faults and security vulnerability, undesired interaction in HCSs have to be controlled. Their detection is difficult due to the large configuration space. |
| HC10  | "Characterizing Complexity of Highly-Configurable Systems with Variational Call Graphs" | The exponentially large configuration space creates various security challenges and threats for development and maintenance of HCSs. |
| HC11  | "FEVER: Feature-oriented Changes and Artefact Coevolution in HCSs"   | HCS’s evolution is a challenging task of knowledge for options to configure all relations and to implement in several kinds of artefacts which needed to ignore the errors of compilation, incorrect products, and invalid code. Most recently conducted studies depend upon the manual analyzed commits, like the job does not make the meaning that can get the quantitative information about a frequency for desired figures nor amount of a exhaustiveness for given figures of evolution for the system of large scale. |
| HC12  | "Test Optimization for Highly-Configurable Cyber-Physical Systems"  | Cybersecurity systems’ testing is not easy to demands a lot of time to verify all the process physically in testing; it will take more time if we add HCS in cyber security because several variants need to test. |
| HC13  | "Feature-Model Interfaces: The Highway to Compositional Analyses of Highly-Configurable Systems" | Software systems are customized through time load or at the time of compilation and options of configuration which are not independent typically, dependencies of these options can only be verified through some models of features, that is a challenging task to maintain and utilized that feature models. |
| HC 14 | “Towards the Automatic Generation and Management of Plant Models for the Validation of Highly-configurable Cyber-Physical Systems” | The plant model simulates the system which is accessed through an embedded system which may have easy activities of verification and validation but manually developing plant models for cyber system physically of several versions of scanning required a lot of time and error free. |
| HC 15 | “A configurable integrated monitoring system for mobile devices” | Smartphone and mobile devices monitoring is a challenging task due to battery power issue and the other computer resources. |
| HC 16 | “A Self-Adaptive Framework for Failure Avoidance in Configurable Software” | To configure that testing is error free is very hard because it’s the manifest that is failures in process because many failures depend on the process of configuration. Their presence will be due to certain feature’s configuration combinations but no other one. |
| HC 17 | “MiL Testing of Highly-configurable Continuous Controllers: Scalable Search Using Surrogate Models” | Software embedded systems like cars facing an issue of testing and verification that is major challenge in the industry of automotive. To consider different hardware configurations of the software to be developed becomes a big issue for engineers. |
| HC 18 | “Improving Preference recommendation and customization In Real World Highly-configurable Software systems” | Software systems of highly configuration have big amount of preferences that is customised by the user. To identify which option of configuration will be update for the behaviour of a particular system is a challenge itself, it is necessary for a user, a tester or debuggers that to look for hundreds or thousands options and all its documentation. |
| HC 19 | “Transfer Learning for Improving Model Predictions in Highly-configurable Software” | Software systems configuration and performance is an issue due to the configuration of changing of environment. It is needed to developed models for the prediction of performance of configurations that is unseen. |
| HC 20 | “A Model-Based Testing Methodology for the Systematic Validation of Highly-configurable Cyber-Physical Systems” | The Cyber Physical Systems become difficult systems due to the combination of the technologies of digital and physically process, it is important to control the changeability in different aspects that is makes the system the complex which leads issues in the configuration of all phases of testing. |
| HC 21 | “A Maneuverable, Configurable, Low-Cost Sensor Deployment Platform” | With the quick growth of sensor technology, sensors are developed in the environments of harmful because of risky human operators or automated robotic solution expenses. |
| HC 22 | “Variability Bugs in HCSs: A Qualitative Analysis” | It is difficult for testers to understand the complex bugs because artefacts are not available, but an error reports that there is a small information about how these errors are produced by the interaction of feature of the big system |
| HC 23 | “Time-space efficient regression testing for configurable systems” | To configure the testing systems are challenging due to number of possibilities of configuration that covered the bugs. Testing becomes more challenging in evolution. |
| HC 24 | “Detecting Control-Flow and Performance Interactions in Highly-Configurable Systems” | Interactions of features distinguish the impact of the availability of one feature by the other feature. To get the information of interactions of features able the predictions based on the behaviour of the performance by the software variations of selected set of different features. |
| HC 25 | “Modelling and Analyzing Highly-Configurable Services” | Due to complexity and variability of services delivers, it is hard to configure the services analysis which is more desirable task; to automate the operation analysis as checking of validity or configuration selection HCS is required. |
| HC 26 | “Mastering Variability Challenges in Linux and Related Highly-Configurable System Software” | More than12,000 configuration options are available in Linux; which leads to issues in development one is of declaration of variability of configuration tools and the other is that what is implemented in the code in real which is to be kept in sync the second option which makes it challenging for static tools analysis. |
| HC 27 | “A Framework for Quantitative Modeling and Analysis of Highly (Re)-Configurable Systems” | The SPL becomes complex in case of modelling and analysis for feature based variability in SDLC due to their unambiguous initiation and organization. |
| HC 28 | “The design and implementation of a configurable MIMO detection system on the NOC-based multicore platform” | An efficient MIMO detector detects the functioning and complexity of the system more well because of its dealing with sizable matrices and vectors arithmetic operations. |
| HC 29 | “Multi-objective test case prioritization in HCSs: A case study” | The prioritization of test cases is highly appreciated. Several techniques are implemented for the achievement of said task. The application of this method is appreciated in HCSs where, a challenge could be the dealing with sizeable configurations. |
| HC 30 | “Enabling Testing of Large Scale HCSs with Search-based Software Engineering: The Case of Model-based Software Product Lines” | Modelling customized SPLs, where testing is critical, by considering the configuration automation, implementation and administration. |
| HC 31 | “A change-aware per-file analysis to compile configurable systems with #ifdefs” | HCSs mostly utilize #ifdefs for system variations as it is hard to efficiently create and compile majority of the configurations. |
|-------|---------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| HC 32 | “A highly versatile and easily configurable system for plant electrophysiology” | This HCS is particular for agricultural domain focusing on secure tailored configurations for measuring several factors. |
| HC 33 | “Analysis Strategies for Configurable Systems” | Sizeable variants for HCSs make their analysis costly. |
| HC 34 | “Test Case Prioritization in Highly-Configurable Systems” | Test case prioritization in HCSs is highly appreciated as sizeable testcases may lead to millions and billions even which is logically not viable. |
| HC 35 | “A Scalable Workflow for a Configurable Neuromorphic Platform” | Mapping sizable neural stages with conventional frameworks speedily helps inviable in view of the basic computational varieties confining the scope and complications. |
| HC 36 | “Intelligent supporting techniques for the maintenance of constraint-based configuration systems (CBCS)” | CBCS focuses on customer involvement and lessen the number of legitimate products. Intelligent supporting techniques helps in performing maintenance task efficiently and effectively. |
| HC 37 | “Tool Demo: Testing Configurable Systems with FeatureIDE” | Value based tailored configured systems are very hard to test. Single configuration may be workable however, multiple configurations may have tangleness and scatteredness issues. |
| HC 38 | “SPLat: Lightweight Dynamic Analysis for Reducing Combinatorics in Testing Configurable Systems” | A thorough testing of SPL based systems are costly due to their testing several combinations of same product. Several test cases against each configuration combinations are required. |
| HC 39 | “PrefFinder: Getting the Right Preference in Configurable Software Systems” | The preferences in HCSs are usually sizeable which a user desire to have, however, their document might be distributed somewhere. Implementing the valuable preferences required millions of documents search for the modification of a particular system behaviour. |
| HC 40 | “Testing of Configurable Systems” | Huge extra effort is required for the testing purpose of HCSs, as, they need different combinations of same testcase to run on the same system. |
| HC 41 | “Method and System for Automatic Documentation of Configurable Systems” | Documentation of every instance is highly appreciated because of the aid of troubleshooting and maintenance purpose. However, it is not an easy task, requires subject matter experts. |

**APPENDIX B**

| S. No | Name |
|-------|------|
| RP 1  | “Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products” |
| RP 2  | “A Framework for Prioritization of Quality Requirements for Inclusion in A Software Project” |
| RP 3  | “Value-Based Requirements Prioritization: Usage Experiences” |
| RP 4  | “The Art of Requirements Triage” |
| RP 5  | “Towards Automated Requirements Triage” |
| RP 6  | “Towards Automated Requirements Prioritization and Triage” |
| RP 7  | “Cognitive-Driven Requirements Prioritization: A Case Study” |
| RP 8  | “Binary Priority List for Prioritizing Software Requirements” |
| RP 9  | “A Cost-Value Approach For Prioritizing Requirements” |
| RP 10 | “Rating Scales and Prioritization In QFD” |
| RP 11 | “An Agent-Oriented Approach to Requirements Engineering” |
| RP 12 | “Facing Scalability Issues in Requirements Prioritization with Machine Learning Techniques” |
| RP 13 | “A Quality-Based Requirement Prioritization Framework Using Binary Inputs” |
| RP 14 | “Requirement Acquisition, Analysis, And Synthesis in Quality Function Deployment” |
| RP 15 | “New Blancharer Theory for Requirements Prioritization” |
| RP 16 | “Towards A Research Framework on Requirements Prioritization” |
| RP 17 | “Requirements Prioritization Based on Benefit And Cost Prediction: A Method Classification Framework” |
| RP 18 | “Decision-Theoretic Requirements Prioritization: A Two-Step Approach for Sliding Towards Value Realization” |
REFERENCES

[1] A. Von Rhein, A. Grebahn, S. Apel, N. Siegmund, D. Beyer, and T. Berger, “Presence-condition simplification in highly-configurable systems,” in Proc. Int. Conf. Softw. Eng., vol. 1, pp. 178–188, May 2015.

[2] R. Schröter, S. Krieter, T. Thüm, F. Benduhn, and G. Saake, “Feature-model interfaces: The highway to compositional analyses of highly-configurable systems,” in Proc. 38th Int. Conf. Softw. Eng., 2016, pp. 667–678.

[3] S. Nadi, T. Berger, C. Kästner, and K. Czarnecki, “Mining configuration constraints: Static analyses and empirical results,” in Proc. 36th Int. Conf. Softw. Eng. (ICSE), 2014, pp. 140–151.

[4] U. Vyas, Applied OpenStack Design Patterns: Design Solutions for Production-Ready Infrastructure With OpenStack Components. New York, NY, USA: Apress, 2016.

[5] M. Cohen, M. Dwyer, and J. Shi, “Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach,” IEEE Trans. Softw. Eng., vol. 34, no. 5, pp. 633–650, Sep. 2008.

[6] B. Nuseibeh and S. Easterbrook, “Requirements engineering: A roadmap,” in Proc. Conf. Future Softw. Eng., 2000, pp. 35–46.

[7] A. B. S. Jerez, “Test case prioritization in highly-configurable systems,” Ph.D. dissertation, Dept. de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Seville, Spain, 2016.

[8] D. Marijan and S. Sen, “Detecting and reducing redundancy in software testing for highly-configurable systems,” in Proc. IEEE 18th Int. Symp. High Assurance Syst. Eng. (HASE), Jan. 2017, pp. 96–99.

[9] O. Stecklina, S. Kornemann, and A. Krumholz, “langOS-A low power application-specific configurable operating system,” in Proc. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN), Potsdam, Germany, 2014, pp. 9–12.

[10] N. Siegmund, A. Grebahn, S. Apel, and C. Kästner, “Performance–influence models for highly-configurable systems,” in Proc. 10th Joint Meeting Found. Softw. Eng., 2015, pp. 284–294.

[11] X. Han and T. Yu, “An empirical study on performance bugs for highly-configurable software systems,” in Proc. 10th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas., Sep. 2016, p. 23.
A. Arrieta, G. Sagardui, and L. Etxeberria, "A model-based testing approach for configurable software," in Proc. IEEE/ACM Int. Conf. Softw. Eng. Companion (ICSE-C), May 2016, pp. 668–670.

C. Kästner and J. Pfeffer, "Limiting recertification in highly-configurable systems: Analyzing interactions and isolation among configuration options," in Proc. Symp. Bootcamp Sci. Secur., 2014, p. 23.

T. Nguyen, U. Koc, J. Cheng, J. S. Foster, and A. A. Porter, "Gens: Dynamic interaction inference for configurable software," in Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2016, pp. 655–665.

J. Meinicke, C.-P. Wong, C. Kästner, T. Thüm, and G. Saake, "On essential configuration complexity: Measuring interactions in highly-configurable systems," in Proc. 31st IEEE/ACM Int. Conf. Automated Softw. Eng., 2016, pp. 483–494.

G. Ferreira, C. Kästner, J. Pfeffer, and S. Apel, "Characterizing complexity of highly-configurable systems with variational call graphs: Analyzing configuration options interactions complexity in function calls," in Proc. Symp. Bootcamp Sci. Secur., 2015, p. 17.

N. Dintzner, A. Van Deusen, and M. Pinzger, "FEVER: An approach to analyze feature-oriented changes and artefact co-evolution in highly configurable systems," Empirical Softw. Eng., vol. 23, no. 2, pp. 905–952, Apr. 2018.

U. Markiegi, "Optimisation for highly-configurable cyber-physical systems," in Proc. 21st Int. Symp. Softw. Product Line Conf., 2017, pp. 139–144.

A. Ariete, G. Sagardui, and L. Etcheberria, "Towards the automatic generation and management of plant models for the validation of highly-configurable cyber-physical systems," in Proc. IEEE Emerg. Technol. Factory Autom. (ETFA), Sep. 2014, pp. 1–8.

C. Miller and C. Poellabauer, "Configurable integrated monitoring system for mobile devices," Procedia Comput. Sci., vol. 34, pp. 410–417, Aug. 2014.

J. Swanson, "A self-adaptive framework for failure avoidance in configurable software," Univ. Nebraska-Lincoln, Lincoln, NE, USA, Tech. Rep. 69, 2014. [Online]. Available: https://digitalcommons.unl.edu/computerscids/69/

R. Matimejed, S. Nejati, L. Briand, and T. Bruckmann, "ML testing of highly configurable continuous controllers: Scalable search using surrogate models," in Proc. 29th ACM/IEEE Int. Conf. Automated Softw. Eng., 2014, pp. 163–174.

D. Jin, "Improving preference recommendation and customization in real world highly-configurable software systems," Univ. Nebraska-Lincoln, Lincoln, NE, USA, Tech. Rep. 84, 2014. [Online]. Available: https://digitalcommons.unl.edu/computerscids/84/

P. Jamshidi, M. Velez, C. Kästner, N. Siegmund, and P. K Hawthikey, "Towards learning for improving model predictions in highly-configurable software," in Proc. 12th Int. Symp. Softw. Eng. Adapt. Self-Manag. Syst., May 2017, pp. 31–41.

A. Ariete, G. Sagardui, and L. Etcheberria, "A model-based testing methodology for the systematic validation of highly-configurable cyber-physical systems," in Proc. 6th Int. Conf. Adv. Syst. Test. Validation Life-cycle, 2014, pp. 56–72.

D. A. Carnegie and J. McVay, "A maneuverable, configurable, low cost sensor deployment platform," in Proc. IEEE Int. Conf. Intell. Meas. Technol. Conf. (I2MTC), May 2014, pp. 1399–1404.

I. Abal, J. Melo, S. Stanculescu, C. Brabrand, M. Ribeiro, and A. Wąsowski, "Variability bugs in highly configurable systems: A qualitative analysis," Trans. Softw. Eng. Methodol., vol. 26, no. 3, pp. 1–34, Jan. 2018.

S. Souto and M. d’Amorim, "Time-space efficient regression testing for configurable systems," J. Syst. Softw., vol. 137, pp. 733–746, Mar. 2018.

A. Denk, "Detecting control-flow and performance interactions in highly-configurable systems," M.S. thesis, Univ. Passau, Passau, Germany, 2017. [Online]. Available: https://www.infosun.fim.uni-passau.de/spl/volltextserver/17190/

J. García-Galán, J. M. García, P. Trinidad, and P. Fernández, "Modelling configuration complexity: Measuring interactions in highly-configurable systems," in Proc. 21st Int. Conf. Adv. Syst. Test. Validation Life-cycle, 2014, pp. 67–78.

M. Al-Hajjaji, J. Meinicke, S. Krieter, R. Schröter, T. Thüm, and G. Saake, "Tool demo: Testing configurable systems with FeatureIDE," SIGPLAN Notes, vol. 52, no. 3, pp. 173–177, Oct. 2016.

J. S. Foster, A. Valladolid, and J. S. Foster, "A scalable workflow for a configurable neurorrhaphy platform," Ph.D. dissertation, Ruperto Carola Univ. Heidelberg, Berlin, Germany, 2014. [Online]. Available: https://archiv.ub.uni-heidelberg.de/volltextserver/17190/

M. Al-Hajjaji, J. Meinicke, S. Krieter, R. Schröter, T. Thüm, and G. Saake, "Tool demo: Testing configurable systems with FeatureIDE," SIGPLAN Notes, vol. 52, no. 3, pp. 173–177, Oct. 2016.

C. H. P. Kim, "Splat: Lightweight dynamic analysis for reducing combinatorics in testing configurable systems," in Proc. 9th Joint Meeting Found. Softw. Eng., 2013, pp. 257–267.

J. Swanson, "Towards the automatic generation and management of plant models for the validation of highly-configurable cyber-physical systems," in Proc. 29th ACM/IEEE Int. Conf. Automated Softw. Eng., 2014, pp. 151–162.

A. Ali, "Testing of configurable systems," in Advances in Computers, vol. 89. Amsterdam, The Netherlands: Elsevier, 2014, pp. 131–140.

A. Bakman, D. Sabin, T. Hubulei, and S. Wertsbergh, "Method and system for automatic documentation of configurable systems," U.S. Patent 9959 115 B2, May 1, 2019.

Z. Racheva, M. Daneva, and L. Buglione, "Supporting the dynamic prioritization of requirements in agile development of software products," in Proc. 2nd Int. Workshop Softw. Product Manage. (WSPM), 2008, pp. 49–58.

R. Thakurta, "A framework for prioritization of quality requirements for inclusion in a software product," Softw. Qual. J., vol. 21, no. 4, pp. 573–597, 2013.

N. Kukreja, S. S. Payyavula, B. Boehm, and S. Padmanabhuni, "Value-based requirements prioritization: Usage experiences," Procedia Comput. Sci., vol. 16, pp. 806–813, Feb. 2013.

A. Davis, "The art of requirements triage," Computer, vol. 36, no. 3, pp. 42–49, Mar. 2003.

P. Laurent, J. Cleland-Huang, and C. Duan, "Towards automated requirements triage," in Proc. 15th IEEE Int.Req. Eng. Conf. (RE), Oct. 2007, pp. 131–140.

C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski, "Towards automated requirements prioritization and triage," Requirements Eng., vol. 14, no. 2, pp. 73–89, 2009.

N. M. Carod and A. Cechich, "Cognitive-driven requirements prioritization: A case study," in Proc. 9th IEEE Int. Conf. Cognit. Informat. (ICCI), Jul. 2010, pp. 75–82.

T. Bubensée, I. van de Weerd, and S. Brinkkemper, "Binary priority list for prioritizing software requirements," in Proc. Int. Work. Conf. Require- ments Eng. Found. Softw. Quality, 2010, pp. 67–78.

J. Karlsson and K. Ryan, "A value-cost approach for prioritizing requirements," IEEE Softw., vol. 14, no. 5, pp. 67–74, Sep./Oct. 1997.

F. Franceschini and A. Rupil, "Rating scales and prioritization in QFD," Int. J. Qual. Rel. Manage., vol. 16, no. 1, pp. 85–97, Feb. 1999.

V. Gaur, A. Soni, and P. Bedi, "An agent-oriented approach to require- ments engineering," in Proc. IEEE 2nd Int. Adv. Comput. Conf. (IACC), Feb. 2010, pp. 449–454, 2010.

P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Facing scalability issues in requirements prioritization with machine learning techniques," in Proc. 13th IEEE Int. Conf. Requirements Eng., Aug. 2005, pp. 297–305.
M. Ramzan, M. A. Jaffar, and A. A. Shahid, “Value based intelligent requirements prioritization framework using binary inputs,” in Proc. AMS Asia Modeling Symp. 4th Int. Conf. Math. Modeling Comput. Simulation, 2010, pp. 187–192.

X. F. Liu, K. Noguchi, and W. Zhou, “Requirement acquisition, analysis, and synthesis in quality function deployment,” Concurrent Eng., vol. 9, no. 1, pp. 24–36, 2001.

T. M. Feilmann, “New Lanchester theory for requirements prioritization,” in Proc. 2nd Int. Workshop Softw. Prod. Manag. (ISWPM), vol. 8, 2008, pp. 1–6.

P. Berander, K. A. Khan, and L. Lehtola, “Towards a research framework on requirements prioritization,” in Proc. 6th Conf. Softw. Eng. Res. Pract., Umeå, Sweden, Oct. 2006, pp. 39–48.

D. M. Berry, “The importance of ignorance in requirements engineering,” J. Syst. Softw., vol. 28, no. 2, pp. 179–184, Feb. 1995.

Z. Racheva, M. Daneva, A. Herrmann, and R. J. Wieringa, “A conceptual model and process for client-driven agile requirements prioritization,” in Proc. 5th Int. Conf. Res. Challenges Inf. Sci., 2010, pp. 287–298.

K. Wiegers, “First things first: Prioritizing requirements,” Softw. Develop., vol. 7, no. 9, pp. 48–53, 1999.

A. K. Massey, P. N. Otto, and A. I. Antón, “Prioritizing legal requirements,” in Proc. 2nd Int. Workshop Requirements Eng. Law (RELAW), 2009, vol. 1936, no. 111, pp. 27–32.

N. R. Mead, “Requirements prioritization introduction,” in Proc. 2nd Int. Conf. Emerg. Trends Eng. Technol. (ICETET), pp. 417–420.

M. Sahlberg and A. Karasira, “A study on the importance of order fit and cost prediction: A method classification framework,” in Proc. 54th EUROMICRO Conf. Softw. Eng. Adv. Appl. (SEAA), 2008, pp. 240–247.

N. Kukreja, “Decision theoretic requirements prioritization: A two-step approach for sliding towards value realization,” in Proc. Int. Conf. Softw. Eng., 2013, pp. 1465–1467.

M. A. Jaffar, “Quality function deployment for large systems,” NASA Langley Res. Center, Hampton, VA, USA, 2002.

YASER HAFEZ

received the Ph.D. degree from International Islamic University Islamabad, Pakistan. He is currently working as an Associate Professor at University Institute of Information Technology, PMAS-Arid Agriculture University Rawalpindi, Pakistan. His research interests are in software engineering and knowledge management.

ATIF ALI is currently pursuing the Ph.D. degree in CS with PMAS Arid Agriculture University. He has been an IT Professional, since 2008. His research interest includes software engineering, requirement prioritization, and highly configurable systems.

[Online]. Available: https://ntrs.nasa.gov/search.jsp?R=20040129650
SHARIQ HUSSAIN received the master’s degree in computer science from PMAS Arid Agriculture University, Rawalpindi, Pakistan, in 2007, and the Ph.D. degree in applied computer technology from the University of Science and Technology Beijing, Beijing, China, in 2014. Since 2014, he has been with the Department of Software Engineering, Foundation University Islamabad, Rawalpindi Campus, where he is currently an Assistant Professor. His main research interests include web services, QoS in web services, web service testing, the IoT, context awareness, and e-learning.

SHUNKUN YANG received the B.S., M.S., and Ph.D. degrees from the School of Reliability and Systems Engineering, Beihang University, in 2000, 2003, and 2011, respectively. He was also an Associate Research Scientist with Columbia University, from 2014 to 2015. He has been an Associate Research Professor with Beihang University, since 2016. His main research interests are reliability, testing and diagnosis for embedded software, CPS, the IoT, intelligent manufacturing, and so on.

***