Three-step Decision Framework for Planning Software Releases

José del Sagrado
University of Almería
Almería, Spain
email: jsagrado@ual.es

Isabel M. del Águila
University of Almería
Almería, Spain
email: imaguila@ual.es

Alfonso Bosch
University of Almería
Almería, Spain
email: abosch@ual.es

***Abstract*—We propose a framework that connects the previously solved problems, which are involved in the setting of the next release goal. A complete workflow has been defined in order to manage the need to properly set the release goal when different, conflicting stakeholders define numerous requirements. Since they cannot all be satisfied by the available resources it necessary to reach an agreement that can be supported by our framework.**

***Keywords*—stakeholder identification; next release problem; requirements negotiation.**

I. Introduction

Today, systems are no longer isolated local applications; they are large and complex systems with an increasing number of connections to other similar applications. These large-scale software systems are developed worldwide, involving teams of software developers, designers, testers, project managers, and other stakeholders working together to deliver a software solution that meets specific requirements [1].

In this context, requirement engineering is not only about capturing and managing requirements but also about fostering collaboration, responding to evolving needs, and adapting to changing project circumstances in order to deliver a successful software solution [2]. It focuses on understanding and defining the needs and expectations of the software system being developed, taking into account multiple stakeholders and a wide range of functionality, expressed and modelled as requirements that may or may not be included in the product being developed.

In addition, due to the limited resources available for the next release of the current project, not all stakeholders' requests can be included in the next product to be delivered, and some will be left for later releases. At this point, software development teams need to manage and review data from multiple sources and make decisions based on the risk associated with each requirement, the cost of delivering it, the benefits the candidate will provide, or some other issues. Timelines, dependencies, resource constraints, and other factors affect this management task [3].

All these factors are estimated or assessed, usually subjectively, by a large group of stakeholders who affect or will be affected by the software under construction. We consider these stakeholders as a source of data to be managed in order to obtain the best set of requirements according to the various criteria defined for the project. However, the best solution is not always the one chosen to maximize objectives, and some kind of agreement, sometimes negotiation, is required [3]. Therefore, three questions should be considered:

- Who assesses the attributes of the requirements?
- What is the best set of requirements?
- Do we have an agreement to build the release?

In this paper, we propose a framework to answer these three questions (see Figure 1). Each of the processes involved in the workflow shown can be treated as a separate problem. The identification of stakeholders, the selection of requirement sets to be included in the next software release, and the process of reaching agreement on the release goal. The three stages are problems that have been previously studied and for which separate solutions have been proposed. Our contribution is the framework that connects all three earlier solved problems, defining a complete workflow to manage the need to properly define the release goal when different, conflicting stakeholders define numerous requirements that cannot all be covered by the available resources.

The remainder of the paper is structured as follows. Section II presents the architecture of the proposal, including the description of the three stages: stakeholder identification, elicitation of candidate requirement sets, and the next release. In Section III, these stages are applied to a case study. A discussion of the limitations and scope of the framework, including what we add to previous proposals, is included in Section IV. Finally, Section V includes the conclusion and future work,

II. Software Release Planning Framework

To address the three issues raised, the framework is divided into three phases or stages (see Figure 1). The first phase (*stakeholder identification*) deals with the identification of the relevant stakeholders in the software project who will be taken into account when proposing the requirements that will be used to define the next release goal. In the second phase (*elicitation of candidate requirement sets*), the requirements proposed by the relevant stakeholders are collected and, based on their assessments, an optimization problem is defined, from which different alternatives (i.e., candidate requirement sets) for the next release are obtained. Finally, in the third phase (*next release agreement*), the aim is to reach agreement on the set of requirements that will make up the next version of the software, selecting one of the candidate sets of requirements found in the previous stage on the basis of productivity indicators and the degree to which stakeholder suggestions have been taken into account,

 ISBN: 978-1-68558-098-8

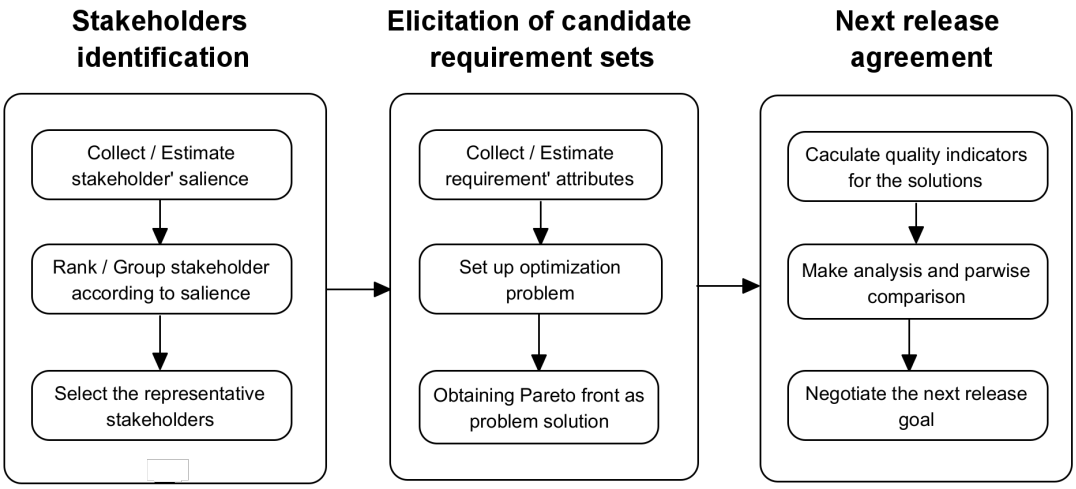

Figure. 1. Workflow defined for the framework.

A. Stakeholders Identification

Stakeholder identification ensures that all individuals, groups, or organisations with a valid interest in the project are considered, as they are the main source of requirements. This is even more evident in large projects where there is a huge community of stakeholders to consider. Their demands are likely to be diverse and conflicting.

An alternative is to reduce the number of stakeholders to manage, but maintain stakeholder coverage [4]. Identifying key stakeholders would reduce the effort required to define the release goal by focusing on the most influential stakeholder representative.

Let $\mathbf{Stk} = \{sk_1, sk_2, \ldots, sk_q\}$ be the set of stakeholders to consider. They represent candidate stakeholders who may be involved in the definition of new features for the next version of a given product.

Each stakeholder is characterised by its salience based on *power*, *legitimacy* and *urgency* as salience components [5]. These values are revealed, usually through interviews, by people involved in the project, who may or may not be stakeholders. Each interviewee could assign a value to the three salience components, but this is not necessary for all of them, and the sets of interviewees for the components can be disjoint. There are h, k and q interviewees about power, legitimacy and urgency respectively, where wp_{ij}, wl_{ij} and wu_{ij} are the values that the interviewee i gives to the stakeholder j in $\mathbf{Stk}$. The power, legitimacy and urgency of a stakeholder j could be defined by aggregating the values obtained from the interviewees.

$$\begin{aligned} p_j &= \textstyle\sum_{i=1}^{h} wp_{ij}, \\ l_j &= \textstyle\sum_{i=1}^{k} wl_{ij}, \\ u_j &= \textstyle\sum_{i=1}^{q} wu_{ij}. \end{aligned} \tag{1}$$

We can define different strategies to select the most influential stakeholders, for example, by clustering them [4] or giving them a weight according to the number of groups identified [6]. As a result, we have a set of m stakeholders who are allowed to propose the requirements that will define the next release goal.

So we can answer the first question in the affirmative, because this stage, *stakeholders identification*, certainly has the ability to define *who assesses the attributes of the requirements*.

B. Elicitation of Candidate Requirement Sets

Let $\mathbf{R} = \{r_1, r_2, \ldots, r_n\}$ be the set of requirements to be considered. These represent new functionalities of the current system suggested by a set of m stakeholders, $\mathbf{Stk} = \{sk_1, sk_2, \ldots, sk_m\}$. $\mathbf{R}$ represents the candidates for inclusion in the next software release. The stakeholders are responsible for setting the preference value of the requirements by defining a value matrix, where each v_{ij} is the subjective value that the stakeholder $sk_i \in \mathbf{Stk}$ assigns to the requirement r_j. The stakeholders to be considered are those that have been selected in the process described in the previous section, and since they do not all have the same importance to the project, it is also defined and $\mathbf{W} = \{w_1, w_2, \ldots, w_m\}$ as the set of weights representing the importance of stakeholders, these values may or may not be calculated based on the salience components. Thus, for a given requirement $r_j \in \mathbf{R}$, its satisfaction s_j is:

$$s_j = \sum_{i=1}^{m} w_i * v_{ij}. \tag{2}$$

In addition, each r_j has an associated cost e_j, which indicates the development effort required to develop it, as estimated by the developers, resulting in the set $\mathbf{E} = \{e_1, e_2, \ldots, e_n\}$. Each software release has a cost limit B, which represents the amount of available resources that cannot be exceeded.

We are able to formulate an optimization problem to be solved in order to obtain the candidate requirement sets, $\mathbf{U}$, to be included in the next release using Pareto dominance as [7]:

 ISBN: 978-1-68558-098-8

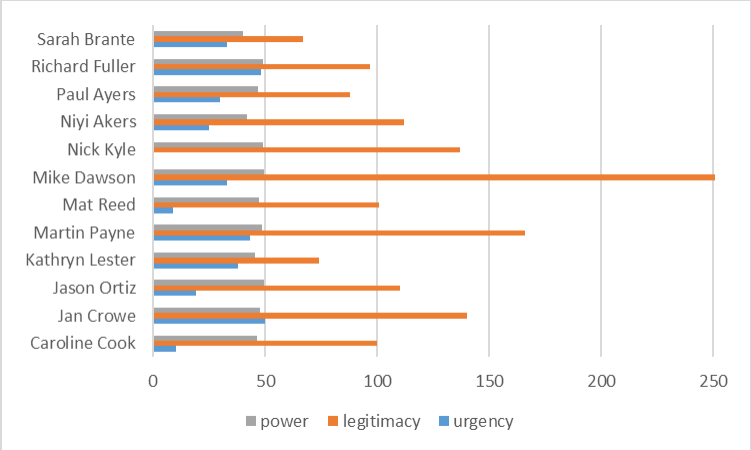

Figure. 2. 12 Stakeholders selected.

$$\begin{array}{ll} \max \sum_{j \in \mathbf{U}} \mathrm{s}_j, & \\ \min \sum_{j \in \mathbf{U}} \mathrm{e}_j, & \\ \text{subject to} & \sum_{j \in \mathbf{U}} \mathrm{e}_j \leq B. \end{array} \tag{3}$$

In addition to the effort-bound constraint (B), alternative formulations may also include constraints associated with requirement relationships or interactions. Specifically, structural interactions impose a particular implementation order, thereby downsizing the NRP feasible solution set [8]. *Implication*, *combination* and *exclusion* dependencies are the functional interactions most commonly studied in the NRP literature:

- Implication interaction, (r_i *implies* r_j), models that a requirement r_i must be implemented before the requirement r_j.
- Combination interaction, (r_i *combined with* r_j), indicates that both requirements must be developed in the same iteration.
- Exclusion interaction. The interaction (r_i *excludes* r_j) reveals that both requirements are incompatible. That is, they could not be developed in the same product.

Next, an optimization algorithm is used to obtain the set of Pareto optimal solutions, which we call candidate requirement sets because all the solutions on the Pareto front are feasible to be considered as the next release goal. There are many solving techniques that have been applied in order to find this set of requirements, including algorithms based on genetic inspiration, the use of nature-inspired optimization, linear programming, clustering approaches or even exact methods for finding the entire Pareto front. A detailed study of their quality is beyond the scope of this paper [9]. Since it is possible to obtain the candidate requirement sets, we can use them to identify the best requirement set, which provides a positive answer to the second research question. Finally, after an analysis of the alternatives obtained, the one to be implemented is chosen by reaching an agreement on the release objective.

C. Next Release Agreement

The task of software release planning does not end when the Pareto front is obtained. To answer the last question, "*Do we have an agreement to build the release?*", the development team must choose which of the alternative sets of requirements (i.e., Pareto optimal solutions) will be implemented.

Due to the black-box nature of optimization algorithms [10], Human experts in charge of decision making need to be supported by additional analysis of optimization results. Although there are well known quality indicators to measure the performance of algorithms [9] and Pareto fronts (such as Hypervolume or Spread), we propose the use of quality indicators that, correctly displayed by some kind of tool, guide decision makers when comparing solutions [11] at the software level. Thus, in addition to the data resulting from the optimization algorithms (such as the number of requirements in a solution, the detailed list of requirements, and the values achieved by the solutions in the objective functions), some other useful indicators can be calculated.

The first is *Productivity*. Let $\mathbf{U} \subseteq \mathbf{R}$ be the solutions under analysis, then

$$\mathrm{prod}(\mathbf{U}) = \mathrm{sat}(\mathbf{U})/\mathrm{eff}(\mathbf{U}), \tag{4}$$

is the benefit obtained by the solution, expressed in terms of how much satisfaction is obtained per unit of effort.

Another indicator is the measure of the amount covered by a solution with respect to everything that is raised by the stakeholder (i.e., stakeholder fairness), which is called *coverage* [11]. Thus, given a stakeholder $sk_i \in \mathbf{Stk}$, this measure associated to a solution $\mathbf{U} \subseteq \mathbf{R}$ with respect to all the requirements valuated by her/him is

$$\mathrm{stcov}_i(\mathbf{U}) = \sum_{j \in \mathbf{U}} v_{ij} / \sum_{j \in \mathbf{R}} v_{ij}, \tag{5}$$

where v_{ij} is the value that the stakeholder sk_i assigns to requirement r_j.

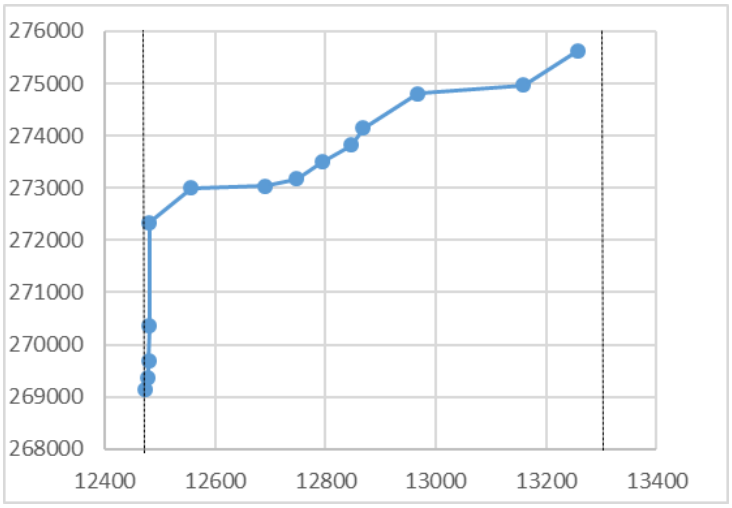

Figure. 3. Pareto front.

III. Case Study

In order to show how this framework can be used in a real-world project, we have included a case study of its application. The dataset used to investigate the validity of the research questions is the *Replacement Access, Library and ID Card project* (RALIC). This was a software project to improve the existing access control system at University College London (UCL). The project combined several UCL access control mechanisms (such as access to the library and fitness centre) into one, eliminating the need for a separate library registration process for UCL ID card holders [12]. This is a widely studied dataset within the domain of Requirements Engineering, and has been used in several works with many different approaches.

RALIC identifies stakeholders by creating a network of recommendations. Each recommender selects a set of other stakeholders, gives them a level of influence on the projec, thus defining a network. The RALIC project involves 144 stakeholders in the network, some of whom only act as recommendees. However, not all recommendees or recommenders have proposed an enhancement or new functionality to be included in the software to be built.

The project includes 138 requirements as increases to the actual access, library, and ID card system, which are arranged in three levels: objectives (10), requirements (48), and specific requirements (104); they are represented by $\mathbf{R} = \{r_1, r_2, \ldots, r_n\}$. Their effort range varies from 4 to 7000 persons-hour. Only 75 RALIC stakeholders use the 100-point method (each stakeholder receives 100 points that can be used to vote for the most important requirements) to prioritise the requirements they are interested in; points_{ij} represents the votes that the stakeholder i assigns to the requirement r_j that can be used to calculate the satisfaction of the requirements. However, stakeholders do not vote at the same level as the three defined ones. These facts translated into the reorganisation of requirements (e.g., requirements that nobody asks for are erased), obtaining for our study 83 requirements, $\mathbf{R} = \{r_1, r_2, \ldots, r_{83}\}$ and their corresponding development efforts in $\mathbf{E} = \{e_1, e_2, \ldots, e_{83}\}$, there are not defined interactions between these requirements.

A. RALIC Stakeholders Identification

The identification of relevant stakeholders can be carried out according to different strategies. In fact, it is defined as a separate problem. In this case, we have applied a clustering approach, using k-means with 4 clusters; a detailed description is beyond the scope of this paper [4]. As a result, the set of 144 stakeholders initially considered is reduced to 12 relevant stakeholders.

The result of this first stage is shown in Figure 2. It also shows the value of the components that define the stakeholder salience, so $\mathbf{Stk} = \{stk_1, stk_2, \ldots, stk_{12}\}$.

B. RALIC Elicitation of Candidate Requirement Sets

From these three sets, **Stk**, **R** and **E**, together with the definition of the requirement satisfaction values (Equation 2), we can define the overall next release problem for RALIC as the following optimisation problem:

$$\begin{array}{ll} \max \sum_{j \in \mathbf{U}} s_j, & \\ \min \sum_{j \in \mathbf{U}} e_j, & \\ \text{subject to} & B_1 \leq \sum_{j \in \mathbf{U}} e_j \leq B_2. \end{array} \tag{6}$$

where $\mathbf{U}$ is a solution (that is, a set of requirements that conforms to the next release). The resource / effort limits are defined in the range $[B_1, B_2]$ which, respectively, corresponds to the 20 % effort required to develop all the requirements for B_1, while B_2 is 25 %. These values have been chosen taking into account the contingency value for effort, that is, an allowance made for the risk that something will not be undertaken with the planned estimated effort. We have decided to define a resource limit interval because, on the one hand, the original data set did not include an upper resource limit and, on the other hand, developers usually discard solutions

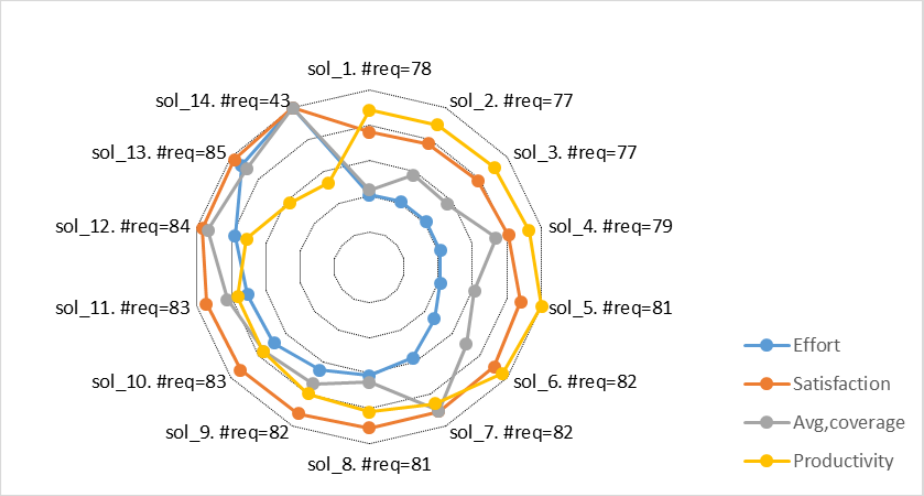

Figure. 4. Quality indicators for the Pareto front solutions.

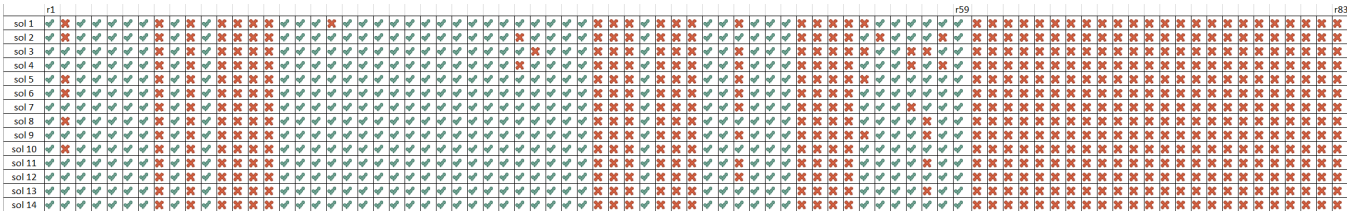

Figure. 5. Distribution of requirements in candidate solutions.

on the Pareto front with low effort value [11]. For RALIC, the values for B_1 and B_2 are 12473.3 and 13304.8, respectively.

To find the Pareto front, we have resorted to a greedy algorithm. This algorithm works as follows: for each effort value in the range $[B1, B2]$, we fill the solution with as many requirements as we can. Then, in a fixed number of attempts, we try to remove each requirement one by one in the solution, replacing it with another that is valid for that level of effort. After each change, the dominance is checked and the solution is replaced if necessary.

Although this may not be the best algorithm for finding Pareto fronts, it is at least simple and produces a Pareto front that can be used to demonstrate the application of the proposed framework. If a better algorithm were used, a more accurate and larger number of candidate solution sets would be found. Figure 3 shows the 14 sets of candidate requirements obtained using this greedy strategy.

C. RALIC Next Release Agreement

The starting point for the final decision is the set of candidate requirement sets, in our case 14 solutions. The analysis includes, on the one hand, the inspection of the values reached in the objective functions (satisfaction, effort) and, on the other hand, the quality indicators defined by the decision makers. For the RALIC case study, these are the number of requirements, productivity, and coverage. It should be noted that we have used the average for the coverage indicator instead of examining the 12 values separately. The use of too many indicators can hinder rather than help decision-making; the general rule should be the simpler the better.

The indicators are shown in Figure 4. One might think that the best candidate solutions are 13 or 14, which achieve the best satisfaction and effort values with 85 and 43 requirements, respectively, and perhaps 13 would be chosen because it covers more requirements. However, when we look at productivity, these alternatives have low values, i.e., using more effort does not result in the same increase in satisfaction.

At this point, the tacit knowledge of developers and stakeholders helps to reach an agreement between possible alternatives. As this is a human process, it is common to use pairwise comparisons of the solutions [11]. For example, let us compare solutions 12 and 7. Both have acceptable satisfaction and productivity values, with a good average stakeholder coverage. The decision in this case could be based on other factors that have not been considered in quantitative terms, such as risk; if we are dealing with higher risk requirements, it is preferable in this case to select solution 7 to have a cushion of effort in case a risky situation needs to be solved.

In some situations, it might be necessary to go down to study the problem at the level of each requirement, negotiating its status in the solution, but this is more difficult to manage ad hoc; just compare Figure 4 and Figure 5, which only show the indicators or the inclusion status for each requirement. It is always better to use quality indicators that are able to summarise the information in a more useful way, just compare

 ISBN: 978-1-68558-098-8

the two figures.

IV. Appraisal of the framework

Previous strategies answer the three questions proposed in this work separately; for example, the problem *"What is the best set of requirements?"* has been formulated as an optimization problem, with customer satisfaction and development cost as the basic optimization objectives, and has been the subject of much research [13]. However, the literature reviewed in this work has neglected the initial identification and prioritization of requirements sources. Similarly, stakeholder identification methods have typically relied on practitioners to manually identify stakeholders based on the use of intuition and experience [14]. Other systematic identification methods follow a set of steps or procedures to ensure consistency, precision, and completeness in achieving the desired result [15], [16]. However, these proposals were not followed by a requirements selection task. Furthermore, the final selection of a solution in the Pareto front could be solved using complex techniques such as ranking of Pareto-optimal solutions or using a mathematical preference model, but no one has connected the three stages in a unique framework, which is precisely our contribution.

V. Conclusion and Future Work

This paper shows how three complex software engineering problems, usually treated independently, are linked together to give a global view of the problem of defining the next release goal for a software product. This framework provides practitioners with a pragmatic approach to solving this complex process in a software engineering project. The three defined stages, *stakeholder identification*, *elicitation of candidate requirement sets*, and *next release agreement*, allow us to manage and improve the tools and/or algorithms defined to solve each stage separately, thus improving the whole process. The validity of our proposal has been demonstrated through its application in a real-world case study, the *Replacement Access, Library and ID Card project* (RALIC) system.

Future work includes the application of the proposed framework to other software projects where data on stakeholders and requirements have been collected. Attention should also be given to investigating the impact on the solutions that make up the NRP solution.

Acknowledgment

This research has been funded by the Spanish Ministry of Science, Innovation and Universities under project PID2019-106758GB-C32 (EML-PA), being also partially supported by the Data, Knowledge, and Software Engineering (DKSE) research group (TIC-181) of the University of Almería.

References

[1] D. Šmite, C. Wohlin, T. Gorschek, and R. Feldt, "Empirical evidence in global software engineering: a systematic review, " Empirical software engineering, vol. 15, pp. 91–118, 2010.

[2] V. Stray and N B. Moe, "Understanding coordination in global software engineering: A mixed, -methods study on the use of meetings and Slack," Journal of Systems and Software, vol. 170, p. 110717, 2020.

[3] K. Brennan (ed.)."A Guide to the Business Analysis Body of Knowledge". IIBA, International Institute of Business Analysis, 2009.

[4] I.M. del Águila and J. del Sagrado, "Salience-based stakeholder selection to maintain stakeholder coverage in solving the next release problem," Information and Software Technology, vol. 160, p. 107231, 2023.

[5] R. K. Mitchell and J. H. Lee, "Stakeholder identification and its importance in the value creating system of stakeholder work," in The cambridge handbook of stakeholer theory, J. S. Harrison, J. B. Barney, R. E. Freeman, and R. A. Phillips, Eds., Cambridge University Press Cambridge, 2019, pp. 53–73.

[6] J. A. Sierra, I. M. del Águila, and J. del Sagrado. "Importance of stakeholders in the next release problem.- Importancia de los interesados en el problema de la siguiente versión," in Actas de las XXV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2021), Abrahão, S. (Ed.), 2021.

[7] Y. Zhang, M. Harman, and S.A. Mansouri, "The multi-objective next release problem," in Proc. of the 9th annual conference companion on Genetic and evolutionary computation, 2007, pp. 1129–1137

[8] J. del Sagrado, I.M. del Águila, and F. Orellana, "Requirements interaction in the next release problem," in Proc. of the 13th annual conference companion on Genetic and evolutionary computation, 2016, pp. 241–242.

[9] J.A. Nuh, T.W. Koh, S. Baharom, M.H. Osman, and S.N. Kew, "Performance Evaluation Metrics for Multi-Objective .Evolutionary Algorithms in Search-Based Software Engineering: Systematic Literature Review", Applied Sciences, vol. 11(7), p. 3117, 2021.

[10] G. Du and R. Guenther, "Two machine-learning techniques for mining solutions of the ReleasePlanner^{tm} decision support system", Information Sciences, vol. 259, pp. 474–489, 2014.

[11] I.M. del Águila and J. del Sagrado, "Three steps multiobjective decision process for software release planning, Complexity vol 21 (S1), pp 250–262, 2016.

[12] S.L. Lim and A. Finkelstein, "Stakerare: using social networks and collaborative filtering for large-scale requirements elicitation," IEEE T Software Eng, vol. 38, pp. 707–735, 2011.

[13] A.M. Pitangueira, R.S.P. Maciel and M. Barros, "Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature," J. Syst. Software, vol. 103, pp. 267–280, 2015.

[14] D. Häuber, K. Lauenroth, H. van Loenhoud, A. Schwarz, and P. Steiger: Handbook ireb certified professional for requirements engineering advanced level elicitation - version 1.0.3. IREB International Requirements Engineering Board, 2019.

[15] L.C. Ballejos and J.M. Montagna, " Method for stakeholder identification in interorganizational environments, Requirements engineering,", vol. 13, pp. 281–297, 2008.

[16] M.M. Rahman, M.M. Moonira and F.T. Zuhora, " A systematic methodology and guidelines for software project manager to identify key stakeholders," International Journal of Research in Computer and Communication Technology, vol. 4, no 8, pp. 509-517, 2015.

.

 ISBN: 978-1-68558-098-8