Investigating Fitness Functions for Search-based Requirements Prioritization

Andrei W. Corezolla
Federal University of Technology - Paraná
Dois Vizinhos - PR, Brazil
andreiwelliton@gmail.com

Francisco Carlos M. Souza
Federal University of Technology - Paraná
Dois Vizinhos - PR, Brazil
franciscosouza@utfpr.edu.br

Lincoln M. Costa
Federal University of Rio de Janeiro
Rio de Janeiro - RJ, Brazil
costa@cos.ufrj.br

Alinne C. Correa Souza
Federal University of Technology - Paraná
Dois Vizinhos - PR, Brazil
alinnesouza@utfpr.edu.br

ABSTRACT

It can be challenging for people to select the most relevant requirement among several software system development options. Requirements prioritization defines the ordering for executing requirements based on their priority or importance concerning stakeholders' viewpoints, which is a problematic task. Based on this problem, this study aims to present a requirements prioritization approach using a genetic algorithm to find optimal solutions, and it can assist in the requirements prioritization activity during the software development process. In this paper, we investigated a set of criteria to create four functions GUT-D, ThS-D, ST, and LT, to assess candidate solutions, i.e., the recommended prioritized requirements. We examine the empirical results concerning the practical approach's effectiveness and cost computational two experiments in the evaluation. We found that the $GUT - D$ fitness function achieved the best fitness value in different settings with 90.51% and 98.63%. Besides that, our study demonstrates that the approach is promising to assist requirements prioritization since each fitness function can be used individually according to companies' necessities.

KEYWORDS

requirements prioritization, search based software engineering, genetic algorithm

1 INTRODUCTION

Quite recently, considerable attention had paid to demands in delivering software products. The competitiveness has encouraged companies to search for solutions to be more agile and efficient to meet customer expectations. Software development depends directly on process, well-defined requirements, and task estimations to ensure the agility of delivers and quality software. Thus, the requirements engineering area provides activities at the start of the development life-cycle to achieve specific attributes such as the cost, complexity, dependency, and importance of each requirement for the project not to exceed the delivery limit or budget.

It is worth mentioning, these attributes above mentioned depend mainly on the domain of software that will be developed. The importance of each requirement is regarding specific stakeholders, i.e., a requirement may afford customer satisfaction but with a high cost for implementation. Therefore, selecting the set of requirements that will compose the development next phase or the new release proves to be a challenge. The main problem is finding the requirements that provide a trade-off between satisfaction customers, budget, and product quality [1].

In requirements engineering, the requirements prioritization is a sub-area that assists the selection requirements activity based on the stakeholder's interests. Requirements prioritization is a process that is managing the importance and urgency of software requirements that will be implemented under constraints of cost, quality, resources, time, and stakeholders' satisfaction [2].

Although the requirements prioritization is already a consolidated process in the software industry and literature, it is considered a complex multi-criteria decision-making process. The complexity occurs mainly for involving different personas such as customer, product owner, project sponsor, developers, and users associated with the difficulty of requirements, team size, and time development. Thus, the problem consists of identifying a set of requirements that maximizes the stakeholder's satisfaction or fulfills specific criteria. Usually, this process is performed manually, but it is subjective once error-prone and labor-expensive. It demonstrates that partial or complete automation has enormous importance to overcome these issues and provide agility in this activity. Nevertheless prioritizing requirements is an activity that cannot be generalized to all companies, an adaptive automatic approach, i.e., that can be able to reuse for different aims is a strategy to agile this task.

Different companies have particular ways of prioritizing their requirements, whether from business value, customer importance, or the significance of the requirement. From this perspective, this study aims to present an approach called *S*earch-Based *R*equirements *P*rioritization (SRP) for requirements prioritization combining aspects as stakeholders interests and requirements attributes such as importance, dependence, and time for development via a search-based algorithm. We performed our approach's evaluation with two independent experiments to verify the effectiveness and computational cost of the SRP approach using the proposed algorithm. Besides, in these experiments, we utilized a software development company's requirements data to analyze the approach adequacy in a real environment.

The remainder of this paper is organized as follows: Section 2 provides the necessary theoretical foundation for the study. Section 3

Formato de Referência: Souza, et al. Investigating fitness functions for search based requirements prioritization. In COMPUTER ON THE BEACH (COTB'21), 12., 2021, Balneário Camboriú. **Anais...** Balneário Camboriú: Universidade do Vale do Itajaí, 2021, p. 1-8.

reports the proposed approach and some related work. Section 4 reports research design of experiments conducted. Section 5 analyzes the results obtained and discusses the proposed approach. Finally, Section 7 makes the concluding remarks and future directions are discussed.

2 BACKGROUND

This section presents the main concepts about Requirement Prioritization, search-based requirement prioritization, and GA.

2.1 Requirement Prioritization

Software requirements commonly undergo several changes in their cycle. Thus, treating and prioritizing them correctly becomes a complex task since it requires a lot of determination and patience, without mentioning the work of dealing with the stakeholders' different interests.

Requirements prioritization is seen as one of the most significant tasks for decision-makers (leaders or managers) within a project [3]. On the other hand, Hudaib et al. [4] treats it as the primary process in software engineering, providing the perfect order of implementation of the requirements to plan software versions and provide desirable functionality to customers.

This prioritization task often occurs given stakeholders' timing or interest, such as a customer who represents more excellent business value for the company, so satisfying its requests has higher priority. However, such requirements can mean something significant only for that customer, and their value-added to the system becomes lower. The literature on requirements prioritization shows various approaches trying to achieve satisfactory results. The GUT matrix and Theme Screening stand out for their simplicity and good performance among several existing prioritization techniques.

2.1.1 GUT Matrix

. The GUT matrix address this complexity and classifies each problem according to Gravity (G), Urgency (U), and Trend (T), generating the acronym (GUT) as detailed below:

- Gravity: It is analyzed considering the intensity or impact that the problem can cause if it is not solved. These damages can be assessed quantitatively or qualitatively.
- Urgency (U): It considers the time frame to solve a given problem. It can be considered urgent problems, deadlines defined by law, or customer response time.
- Tendency (T): It is analyzed by the pattern or trend of the situation evolves. Problems are analyzed considering the development that it will have in the absence of effective action to solve them.

2.1.2 Theme screening

. The Theme Screening technique consists of ordering the functionalities based on business themes used as comparison factors, as illustrated in Fig. 1 [5].

In the Theme screening technique, it is necessary to define the comparison criteria and ensure that all are valid in each requirement. A minimum of five and a maximum of nine criteria shall be defined, where one of them shall be classified as the primary theme and serve as a reference for the others' score. The base theme receives a zero score, while the most important ones receive a positive sign and the less important ones a negative sign.

	Criteria						
	Complexity	Effort	ROI	Integration	Budget	Total	Priority
Differentiated access for subscribers	-	-	+	+	+	+1	2
Make product videos available	-	+	+	-	-	-1	4
Pay by credit card	0	0	0	0	0	0	3
Social media integration	+	+	+	-	+	+2	1
Provide free access	-	-	+	-	+	-1	4

Figure 1: Theme Screening (Adapted from [5])

The value of each prioritized item is obtained by summing the negative and positive signals; at the end, when ordering from the largest to the smallest, we will have the prioritization result.

However, with the projects' intense demands and the complexity of them, traditional techniques of prioritization of requirements lost their effectiveness, bringing a significant problem concerning the prioritization of requirements. In this context, search-based techniques have become a favorable means for prioritizing requirements.

2.2 Search-based Requirement Prioritization

Search-based Software Engineering (SBSE) automates a SE activities utilizing different techniques, such as local and global searches, metaheuristics, and evolutionary algorithms. These techniques offer suitable solutions for complex problems at a reasonable computational cost compared to random techniques. SBSE approaches have been gaining much importance in recent years. It has supported software engineers across a range of software development activities as requirements, code generation, software testing, maintenance, and, requirements prioritization which is the focus of the present paper.

Formally, SBRP may be represented by a set of n requirements to be prioritized, i.e., $R = \{r_1, r_2, ..., r_n\}$. Generally, the requirements come with some attributes that should be considered: i) dependence between them; ii) importance regarding a stakeholder; and iii) complexity. Then, the aim is to find a set of m prioritized requirements, for example, $RP = \{rp_2, rp_4, rp_7..., rp_m\}$ that will be implemented in the next phase of development, or they will enter in a new release of the system [6].

Nevertheless, to address the requirements prioritizing problem in real environments, various aspects should be analyzed because prioritization strategies can variate according to each company. For this reason, an automated approach for RP should cover many interests, be it from the customer, company, or developer. Therefore, it is necessary to adapt and extend traditional techniques to address the real strategies for prioritizing based on interests.

The most common approaches found in the literature are regarding the application of search-based algorithms to the next release problem that uses mainly the satisfaction of customers as criteria [7], [8], [9]. However, nowadays, it does not reflect the companies' reality because they also consider the importance and time to develop each requirement.

So many efforts have been invested in requirements prioritization research. It is possible to find different algorithms as hill climbing, ant colony optimization, and genetic algorithm, one of the most popular. Due to its robustness and flexibility also was a target of our study.

2.3 Genetic Algorithm

Genetic Algorithm (GA) is a search-based technique belonging to evolutionary algorithms classes inspired by Darwinian evolution. GA has been widely recognized as the main search strategy and an optimization method that is often useful for dealing with combinatorial problems and a considerable possibility of solutions [10].

The algorithm works on a set of candidate solutions called population. Each solution (also named individual) is qualified for a specific fitness function. An individual is composed of a chromosome of genes, and each gene represents a piece of information to solute the faced problem.

For the algorithm to find better solutions, genetic operators are known as selection, crossover, and mutation are applied in the population. Selection is the process of selecting parents who recombine themselves to create offsprings for the next generation. The most commonly used selection methods include Roulette Wheel Selection, Rank Selection, and Tournament Selection.

The crossover operator is responsible for reproducing to produce a new population that is fitter than the previous one. Firstly, the parents are selected, and the crossover arises from a modification in each individual. A vector of n-length can represent an individual, and it determines the order in which genes are drawn from parents. After a gene is drawn from one parent and removed from the other, it is added to the offspring chromosome. Lastly, the mutation operator is employed to create small changes in individuals to maintain the diversity of the population [11].

The GA flexibility makes them attractive for many optimization problems in practice. In this context, several studies have demonstrated the GA success in different domains such as music [12], games [13] [14], autonomous driving [15], among others. To achieve this success, the definition of an objective function to quantify a candidate solution is fundamental to guarantee the algorithm finding promising solutions.

TThus, the main component of search-based approaches is the fitness function (also called objective function), which guides the algorithm process towards the search space's promising areas. Thus, more effective functions lead to significantly better results [16] [17]. The fitness function must be defined according to problem features, i.e., the objective is represented by a particular feature capable of evaluating candidate solutions in terms of their goodness and suitability for solving the problem [18].

3 SRP APPROACH

The proposed approach attempts to automate the requirement's prioritization by searching a set of requirements that achieve specific criteria. These criteria are based on traditional prioritization techniques as GUT Matrix and Theme screening. In addition to these techniques, we propose two metrics: customer satisfaction, the dependency between requirements, the importance of stakeholder and requirement, time to develop a requirement, and total hours worked by the developers.

It is worth mentioning that these elements have been defined through a survey with seven project managers and eight requirements analysts of 5 different organizations. One of the main findings was that 66.7% of the participants considered the level of dependencies on requirements and 53.3% the delivery deadlines as fundamental aspects in requirements prioritizing.

Requirement prioritization has been widely investigated as a search problem since it needs to find a set of requirements according to the different elements of a project. Thus, the present study utilizes a Genetic Algorithm combined with four different fitness functions to find the optimal set of prioritized requirements.

As we can see in Figure 2, the process starts from a complete list of input requirements. The GA selects random values for an initial set of prioritized requirements representing the candidate solutions. Then, the GA assesses and improves the solutions iteratively using our fitness functions and the genetic operators. This process is performed to improve fitness value and achieve higher search space coverage until it reaches the total number of generations. Finally, the approach's output is a list containing the prioritized requirements according to a fitness function strategy.

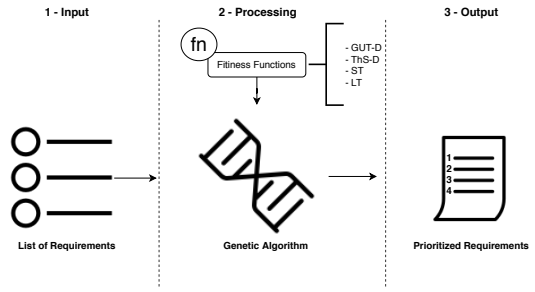

Figure 2: Approach for Requirements Prioritization

Our approach uses the following metrics to represent the proposed fitness functions: (*i*) Adapted GUT Matrix (GUT-D), (*ii*) Adapted Theme Screening (ThS-D), (*iii*) Customers Satisfaction (ST), and the (*iv*) Time for development (LT). These metrics are used to measure the set of requirements' adequacy and direct the search process to an optimum in the search space. The fitness functions work separately into the approach, and they may be used according to the necessity of each company or project.

3.1 Fitness Function $GUT - D$ ($\mathbf{F_1}$)

$GUT - D$ is a function derived from the GUT Matrix, which is a traditional prioritization technique. The GUT-D fitness function consists of three criteria (G, U, and T), also, the Dep criterion that indicates the number of dependencies for a requirement, i.e., how many other requirements it depends on being developed.

For this fitness function, a set of n requirements is assumed, where $R = \{r_1, r_2, ..., r_n\}$. As shown in the equation 1. The $GUT - D$ function consists of the G, U, and T multiplication. The value obtained is penalized with the value of the dependencies of the set

of requirements (Dep). The penalty was defined because the more dependence a requirement has, the longer it will be developed.

$$Maximize \sum_{i=1}^{n} (G_i.U_i.T_i) - Dep \quad (1)$$

Lastly, $GUT - D$ aims to find the set of requirements that maximize the GUT criteria with the least dependencies on requirements. The greater the number of dependencies a requirement has, the worse the group of prioritized requirements.

3.2 Fitness function $ThS - D$ (F_2)

The proposed $ThS - D$ function uses the principles of the theme screening technique to perform the fitness value calculation as presented in Subsection 2.1.2 and the dependency element.

$$Maximize \sum_{i=1}^{n} (Pr_i) - Dep \quad (2)$$

The set of n requirements is assessed by how the value of its priorities is added (Pr). The value is subtracted by the number of dependencies (Dep) of each r_i, thus obtaining the fitness value. Therefore, the $ThS - D$ aims to maximize the priority value obtained through the theme screening variables to find the best set of requirements to be prioritized.

3.3 Fitness function ST (F_3)

The ST function aims to maximize customers satisfaction by selecting n requirements with a lower level of satisfaction, and a longer waiting time in the development queue, i.e., the requirement is essential for the customer, but, due to some technical cause, it may have been judged to be of little relevance, and thus, able to generate dissatisfaction to him. The TS works as computing inversely proportional. The function will evaluate the requirements that are waiting for longer (T), the highest priority (Pr) according to stakeholders, (C) that works with a weight for a customer or stakeholder, and the requirements that generated more dissatisfaction (St).

This computing is obtained by the sum of the values T and Pr. After the result is increased by the value of C of the customer. The result is obtained and later divided by St. The criteria sum is subtracted by the number of dependencies (Dep) from each r_i, thus resulting in the fitness value as we can see in Equation 3:

$$Maximize \sum_{i=1}^{n} \frac{(T_i + Pr_i)^C}{St} - Dep \quad (3)$$

3.4 Fitness Function LT (F_4)

For LT function are selected n requirements with the lower time of development (T), the customer or stakeholders importance (C), the priority of requirements defined by stakeholders (Pr).

The value obtained goes through a constraint assessment. If the value of T of the prioritized requirements is higher than the value of total hours worked by the development team H, the solution is penalized since the requirements are only a few suitable for prioritization. In the Genetic Algorithm, a penalty makes it hard for an individual to move on to the next generation.

$$Maximize \sum_{i=1}^{n} \frac{Pr_i + C}{T_i} \quad (4)$$

In Equation 4, computing the inversely proportional function, the goal is to maximize the fitness value in order to find the best set of prioritized requirements. The lower the value of T, the higher the fitness value.

4 EXPERIMENTAL STUDY

We conducted an experiment to analyze and evaluate the effectiveness of the proposed approach for supporting requirements prioritization using GA. We are interested in measuring the effectiveness in terms of the fitness value and computational cost in time. In this study, the guidelines recommended by Wohlin et al. [19] were used. The experiments were performed through a laptop with Intel Core i7 2.4GHz CPU, 8GB memory in the Linux Ubuntu operating system.

4.1 Experiment Definition

We used the Goal-Question-Metric (GQM) model [20] to set out the objectives of the experiment that can be summarized as follows:

*"Analyze **SRP approach** for the purpose of **evaluation** with respect to **fitness function and computational cost** from the point of view of **experimenters** in the context of **the an organization**."*

For achieving the goal, we seek to investigate the following Research Questions (RQs):

RQ_1: How effective is the SRP approach for requirements prioritization?

In order to answer RQ_1, we compare the fitness value of the each fitness function (F_1, F_2, F_3, F_4). We also performed this experiment 10 times and computed the average (μF_i) of fitness value for each function. We have defined the following hypotheses for this research question:

$H1_0$: There is no difference on fitness value between the GUT-D (μF_1), ThS-D (μF_2), ST (μF_3) and LT (μF_4), thus:

$$H1_0 : \mu F_1 = \mu F_2 = \mu F_3 = \mu F_4$$

$H1_1$: There is difference on fitness value between the GUT-D (μF_1), ThS-D (μF_2), ST (μF_3) and LT (μF_4), thus:

$$H1_1 : \mu F_1 \neq \mu F_2 \neq \mu F_3 \neq \mu F_4$$

RQ_2: How efficient is the SRP approach for requirements prioritization?

The efficiency of the SRP approach was measured using the time for requirements prioritization. We also performed this experiment 10 times and computed the time average (μTF_i). The time was computed in seconds for each fitness function considering three different configurations. The time (T) was computed in seconds and by expression 5.

$$T = T_f - T_s \quad (5)$$

We defined the following hypotheses for this research question:

$H2_0$: There is no difference on time between the GUT-D (μTF_1), ThS-D (μTF_2), ST (μTF_3) and LT (μTF_4), thus:

$$H2_0 : \mu TF_1 = \mu TF_2 = \mu TF_3 = \mu TF_4$$

$H2_1$**:** There is difference on time between the GUT-D (μTF_1), ThS-D (μTF_2), ST (μTF_3) and LT (μTF_4), thus:

$$H2_1 : \mu TF_1 \neq \mu TF_2 \neq \mu TF_3 \neq \mu TF_4$$

4.2 Experiment Design

In this study, 254 maintenance and software evolution requirements (R = $\{r_1, r_2, r_3, .., r_n\}$) from two real bases provided by a software organization were used. The first base contains 41 requirements and the second have 213 requirements. The Table 1 presents the requirements recorded in the database.

Table 1: Requirements recorded in the database.

Req.	Task	Time	Priority	Premium	Dep.	Sat.
1	81146	14:50:26	3	1	0	4.5
2	82187	00:11:30	3	0	0	4.5
3	82187	00:23:05	3	0	1	4.5

The Req. column represents the ID of the requirement. The Task column represents the task to which the requirements belong. The Time column displays the estimated time for the requirement to be met. The priority column represents the requirement's priority level, ranging from 1 to 5, where 1 represents low priority and five high priority. The premium column shows the customer's importance or stakeholder, which has two values, 0 and 1, where 0 indicates that the customer is not very important for the requirements and one indicates that it is Premium (earned value). The Dep. column indicates the number of dependencies on other requirements that prevent this requirement from being met. Finally, the Sat. column shows the level of customer satisfaction concerning the company's services, which ranges from 1 to 5, where 1 represents low satisfaction and five high satisfaction.

We carried out two different experiments ($E = e_1; e_2$). The first experiment (e_1) answered RQ_1 and the second (e_2) was conducted to answer RQ_2. For answering the RQs, this empirical study manipulated an independent variable: fitness function; and four dependent variables were measured:

- **Number of population (*P*):** represents the number of candidate solutions. The population is composed by p individual, where p is represented by three different parameters (P = 10, 20);
- **Mutation Rate (*MR*):** contains two different parameters (MR = 0.3, 0.5);
- **Number of generations (*G*):** represents the number of generations in a GA. We used two different parameters for generation (G = 100, 200).
- **Crossing Rate (*CR*):** contains two different parameters (CR = 0.3, 0.5);

For both experiments, we used two settings ($ST = st_1, st_2$) for each fitness function, which are a combination of the parameters population, mutation rate, generations, and crossing rate respectively. Table 2 present the design overview of these experiments.

4.3 Procedure of Experiment

To answer the RQs, we carried out the experiments as follows: *(i)* requirements set generation; *(ii)* measure fitness value and time; and *(iii)* comparison of the fitness value and time obtained in each fitness function. These experiments were performed in five steps:

Table 2: Experiments Design

Fitness function	Settings	Parameters			
		P	MR	G	CR
F_1, F_2, F_3, F_4	st_1	10	0.3	100	0.3
	st_2	20	0.5	200	0.5

(1) Identifying 254 maintenance and software evolution requirements (R = $\{r_1, r_2, r_3, .., r_n\}$) as experimental subjects.
(2) Prioritizing a set of requirements (*RP*), where $RP \subseteq R$ is generated through AG. RP_n represents the best requirements to be prioritized. For the experiments, $n = 5$ was defined for a better analysis of the prioritized requirements and the database's size.
(3) Computing the fitness value for each *RP* obtained from the four fitness functions;
(4) Comparison between the four fitness on requirements prioritization;
(5) Computing the time in seconds for each fitness function.

5 RESULTS AND DISCUSSION

In this section we answer the RQs presented in Section 4.1 from the analysis of results concerning the effectiveness and efficiency of the proposed approach. The results of the experiment are shown following in separate subsections according to each research question

5.1 Effectiveness of the SRP approach (RQ_1)

In this research question, the fitness value was computed according to each fitness function (F_1, F_2, F_3, F_4) and setting (st_1, st_2).

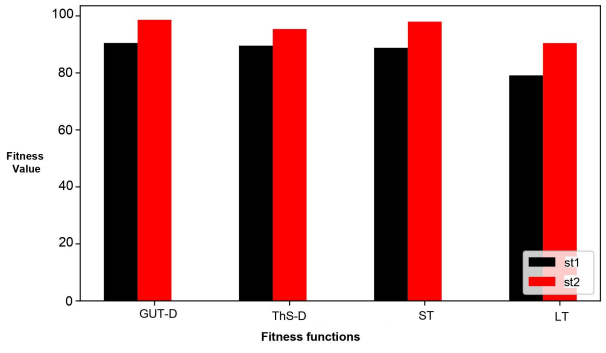

Figure 3: Fitness value achieved by the fitness functions according to each setting.

Figure 3 shows the fitness value on average obtained by the fitness functions according to each setting. The results indicate that the $GUT - D$, on average, achieved the best fitness value in both settings with 90.51% in st_1 and 98.63% in st_2.

The fitness functions $ThS - D$ and ST achieved similar fitness values. $ThS - D$ achieved 89.56% and 95.40% in st_1 and st_2, respectively; while ST achieved 88.81% in st_1 e 97.99% in st_2. The last fitness function, (LT), obtained the lowest fitness value in both settings with 79.14% and 90.47% in st_1 and st_2, respectively. Therefore, the fitness value average obtained in st_2 was 8.49% better than in st_1.

Figures 4, 5, 6 and 7 show the fitness value on average obtained by each fitness function in st_2. The results suggest that the fitness value improved over the 200 generations for st_2. .

Figure 4 presents the best solution found in 200 generations. The result shows that the greatest value, i.e., 100% was achieved in the 30th generation. Figure 5 presents the best solution found in the 108th generation of 200 generations.

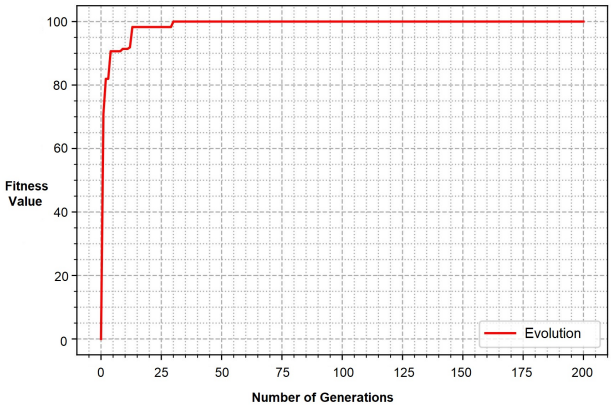

Figure 4: Evolution of GUT-D fitness value in st_2.

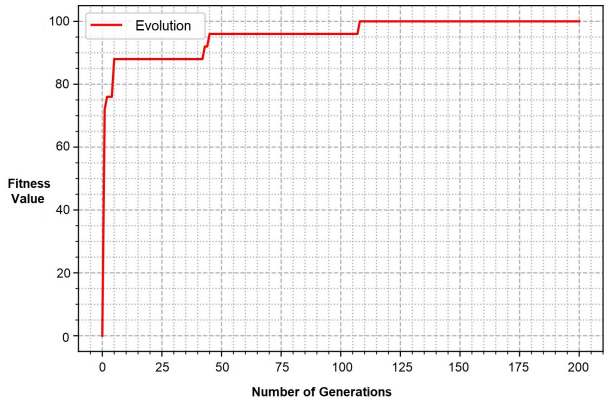

Figure 5: Evolution of ThS-D fitness value in st_2.

From the results presented in Figures 4 and 5, we conducted a comparative analysis of the prioritized requirements using the adapted and traditional techniques, as shown in Tables 3 and 4.

We verified that the functions $GUT - D$ and $TS - D$ presented different results compared to the results of the traditional GUT and TS. This result obtained expected due to the application of the "dependency" criterion on requirements by $GUT - D$ and $TS - D$ that aim to select the requirements based on GA and traditional techniques that generate only by ranking.

Table 3: Requirements prioritization comparison between $GUT - D$ and Matriz GUT.

Techniques	Requirements	Dep	G	U	T	Total
$GUT - D$	6	0	5	5	3	75
	0	0	5	2	5	50
	15	0	3	4	2	28
	1	0	3	3	4	16
	20	0	4	4	1	16
GUT Matriz	30	1	5	5	4	100
	6	0	5	5	3	75
	5	2	5	5	3	75
	0	0	5	2	5	50
	1	0	3	3	4	24

Table 4: Requirements prioritization comparison between ThS-D and TS.

Techniques	Requirements	Priority	Dependency
$ThS - D$	29	5	0
	7	5	0
	21	5	0
	6	5	0
	37	5	0
TS	30	5	1
	6	5	0
	5	5	2
	0	5	0
	16	5	0

For ST and LT functions, the assessment of the sets of requirements occurred separately since these functions were defined based on the features (priority, dependency, the customer or stakeholders importance, and customer satisfaction) of real projects in agile environments.

Figure 6 illustrates the best solution achieved in the 69th generation of 200 generations. The ST function aimed to find the best set of requirements considering the waiting time of the requirements and customer dissatisfaction. Therefore, the ST focused on prioritizing the development requirements for the longest time and those with the least satisfied customers.

In contrast, LT function aimed to identify the best set of requirements taking into account the shortest time. This goal is defined to find requirements with the shortest development time, thus focusing on the agility in delivering the requirements. Finally, Figure 7 presents the best solution achieved by LT function in the 137th generation.

Analyzing the results, we observe that the best fitness value achieved in the $GUT - D$ function was in the 30th generation using the st_2. The $ThS - D$ and ST functions obtained the best fitness value in the 51st and 53rd generations, both using st_1. Finally, the

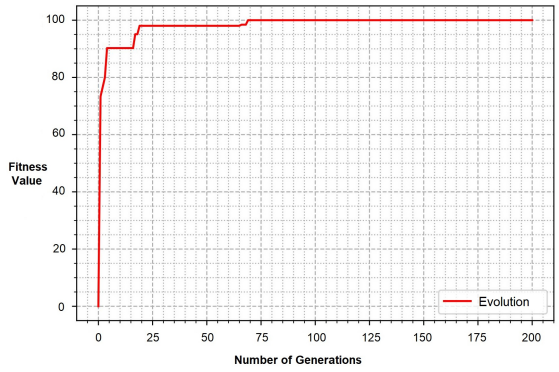

Figure 6: Evolution of ST fitness value in st_2.

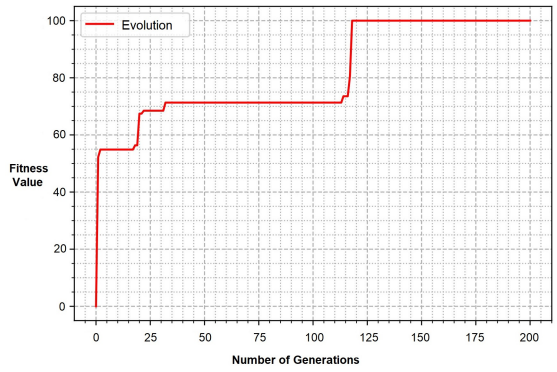

Figure 7: Evolution of LT fitness value in st_2.

LT function was the one that obtained the most discrepant result, reaching its best fitness value in the 75th generation using st_1.

The results analyzed through the Wilcoxon test indicated that there was a significant difference between $GUT-D$, $ThS-D$, ST and LT, which confirmed the alternative hypothesis ($H1_1$), as shown in Table 5.

Table 5: Wilcoxon test for fitness value comparison between fitness function.

Functions	F_1 - F_2	F_1 - F_3	F_1 - F_4	F_2 - F_3	F_3 - F_4	F_4 - F_2
Z	-9,218	-10,185	-6,926	-10,175	-10,165	-9,056
p-value	0.000	0.000	0.000	0.000	0.000	0.000

5.2 Efficiency of the proposed approach (RQ_2)

The second experiment computed the time average of the proposed approach to answering this RQ. Figure 8 reports the results from average time in seconds according to settings for each fitness function.

The results indicate that F_4, on average, obtained the shortest time, 3.06 seconds in st_1 and F_2 with 9.72 seconds in st_2. On the other hand, F_2 was the most time-consuming, with 4.87 seconds in st_1 and F_3 with 14.90 seconds in st_2. Considering all fitness functions,

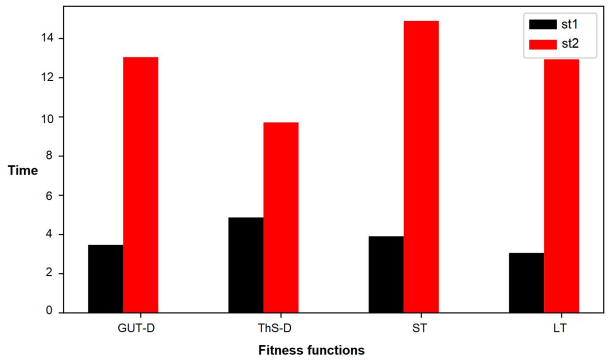

Figure 8: Time achieved by the fitness functions according to each setting.

the average time obtained in st_1 was 3.82, while in st_2 was 12.65 seconds.

As in RQ1, we analyze the results using the Wilcoxon test, as shown in Table 5.2. The results indicate a significant time difference between F_1 - F_2, F_1 - F_3, F_2 - F_3, F_3 - F_4 and F_4 - F_2, which confirmed the alternative hypothesis ($H2_1$). However, the result of the comparison between F_1 - F_4 indicated that there is no difference in time between them, since $p - value > 0.05$ confirming the null hypothesis ($H2_0$).

Table 6: Wilconxon test for time comparison between fitness functions

.

Functions	F_1 - F_2	F_1 - F_3	F_1 - F_4	F_2 - F_3	F_3 - F_4	F_4 - F_2
Z	-2,803	-2,397	-1,886	-2,803	-2,803	-2,803
p-value	0,005	0,017	0,059	0,005	0,005	0,005

6 RELATED WORK

Over the last few years, few publications have appeared in recent years documenting different requirements prioritization techniques. Some of these studies address the use of different techniques such as Artificial bee colony (ABC) [21], Evolutionary Algorithms [22], Non-dominated Sorting Genetic Algorithm II (NSGA-II) [23].

Chaves-González et al. [21] propose a multi-objective swarm intelligence approach called MOBAC, based on an artificial bee colony algorithm, in which several multi-objective features have been included to obtain high-quality results for a realistic Multi-Objective Next Release Problem (MONRP). The MOABC was evaluated through several quality indicators by comparing the results with other approaches published in the literature, such as ACO, NSGA-II, GRASP). The results generated by the approach obtain a hypervolume (HV) of over 60% for the most complex data set managed, while the other approaches published cannot obtain an HV of more than 40% for the same data set.

Kifetew et al. [22] presented a multi-objective formulation of the multi-decision-maker requirements prioritization problem and outlined a solution based on EA. The proposed approach is based on finding Pareto optimal prioritization that exhibits the minimum

levels of disagreement among the various decision-makers involved in the process, thus supporting the decision-maker through a view on alternative optimal requirements prioritization. The ultimate decision-maker selects optimal solutions based on additional interests (e.g., strategic).

The study proposed by Amaral and Elias [23] presented a multi-objective, risk-based approach for the Next Release Problem (NRP), called SR2 (Selection of Requirements based on Software Risks), in which a risk analysis is incorporated for estimating the impact of risks on requirements costs and stakeholders' satisfaction. Based on risk probability and severity, together with the cost of applying mitigation techniques, SR2 estimates the impact of risks on both requirements costs and stakeholders' satisfaction. Experiments with two semi-real datasets have been presented, in which the proposed approach, exploring the NSGA-II algorithm, has obtained a higher number of recommended solutions closer to Pareto and reference fronts.

Although many points were addressed, our study presents an automated approach for requirements prioritization combining aspects as interests of stakeholders and attributes of requirements such as importance, dependence, and time using GA, guided by an objective function.

7 CONCLUSION

We proposed an automated approach for requirements prioritization. Our approach consists of a Genetic Algorithm guided by different fitness functions to prioritize requirements based on traditional prioritization techniques such as GUT Matrix and Theme screening. Besides these techniques, we create two metrics by combining some elements: customer satisfaction, the dependency between requirements, the importance of stakeholder and requirement, time to develop a requirement, and total hours worked by the developers.

Our approach works from 254 maintenance and software evolution requirements from two real bases provided by a software organization. We evaluated our approach through experiments to analyze their effectiveness and efficiency for each fitness function, i.e., $GUT - D$, $ThS - D$, ST, and LT. We carry out two experiments and computed the fitness value achieved and the execution time.

Summing up the results, we noticed that for all experiments, the $GUT - D$ achieved the best fitness value in both settings with 90.51% in st_1 and 98.63% in st_2. However, for the company's context in the experiments, the fitness function that best suited was LT, which employed the importance of the stakeholders, time for the development of that requirement, and level of priority given by stakeholders.

This study demonstrates the feasibility of an intelligent approach for requirements prioritization, in which each fitness function can be used individually according to companies. Therefore, the present work may lead to more robust approaches development to assist in the requirements prioritization process.

Future works are direct towards the following topics: *(i)* a multi-objective approach development to more accurately meet the needs of companies; *(ii)* extends the approach through creating a graphical application that facilitates their manipulation; *(iii)* improvement of the fitness functions; and *(iv)* experiments using different real scenarios.

REFERENCES

[1] Klaus Pohl. *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 3642125778, 9783642125775.

[2] Ashish Sureka. Requirements prioritization and next-release problem under non-additive value conditions. In *2014 23rd Australian Software Engineering Conference*, pages 120–123. IEEE, 2014.

[3] I. Sommerville. *Software Engineering*. Pearson Addison-Wesley, 10^{th} edition, 2016.

[4] Amjad Hudaib, Raja Masadeh, Mais Haj Qasem, and Abdullah Alzaqebah. Requirements prioritization techniques comparison. *Modern Applied Science*, 12, 01 2018. doi: 10.5539/mas.v12n2p62.

[5] Vitor L Massari and André Vidal. *Gestão Ágil de Produtos com Agile Think Business Framework: Guia para certificação EXIN Agile Scrum Product Owner*. Brasport, 2018.

[6] Itzel Morales-Ramirez, Denisse Muñante, Fitsum Meshesha Kifetew, Anna Perini, Angelo Susi, and Alberto Siena. Exploiting user feedback in tool-supported multi-criteria requirements prioritization. In *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, pages 424–429. IEEE Computer Society, 2017. doi: 10.1109/RE.2017.41.

[7] A.J. Bagnall, V.J. Rayward-Smith, and I.M. Whittley. The next release problem. *Information and Software Technology*, 43(14):883 – 890, 2001. ISSN 0950-5849.

[8] P. Baker, M. Harman, K. Steinhofel, and A. Skaliotis. Search based approaches to component selection and prioritization for the next release problem. In *2006 22nd IEEE International Conference on Software Maintenance*, pages 176–185, Sep. 2006.

[9] José Del Sagrado, Isabel María Del Águila, and Francisco Javier Orellana. Ant colony optimization for the next release problem: A comparative study. In *2nd International Symposium on Search Based Software Engineering*, pages 67–76. IEEE, 2010.

[10] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2nd edition, 2003. ISBN 0137903952.

[11] José Fernando Gonçalves, Jorge José de Magalhães Mendes, and Mauricio G C Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1):77–95, 2005. doi: 10.1177/1523422310365309.

[12] M. E. G. Mital, R. Ruzcko N.M.I. Tobias, A. A. Bandala, R. K. Billones, and E. P. Dadios. Utilization of genetic algorithm in classifying filipino and korean music through distinct windowing and perceptual features. In *2019 International Conference on contemporary Computing and Informatics (IC3I)*, pages 121–126, 2019. doi: 10.1109/IC3I46837.2019.9055676.

[13] D. Lessin and S. Risi. Darwin's avatars: A novel combination of gameplay and procedural content generation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO'15, pages 329–336, 2015.

[14] L. M. Costa, A. C. C. Souza, and F. C. M. Souza. An approach for team composition in league of legends using genetic algorithm. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 52–61, 2019. doi: 10.1109/SBGames.2019.00018.

[15] O. A. Yarom, S. Jacobitz, and X. Liu-Henke. Design of genetic algorithms for the simulation-based training of artificial neural networks in the context of automated vehicle guidance. In *2020 19th International Conference on Mechatronics - Mechatronika (ME)*, pages 1–8, 2020. doi: 10.1109/ME49197.2020.9286464.

[16] Leonardo Bottaci. A genetic algorithm fitness function for mutation testing. In *Proceedings of the SEMINALL-workshop at the 23rd international conference on software engineering, Toronto, Canada*, 2001.

[17] Vassilios Petridis, Spyros Kazarlis, and Anastasios Bakirtzis. Varying fitness functions in genetic algorithm constrained optimization: the cutting stock and unit commitment problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(5):629–640, 1998.

[18] K. Dahal, Stephen Remde, P. Cowling, and N. Colledge. Improving metaheuristic performance by evolving a variable fitness function. In *EvoCOP*, 2008.

[19] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen. *Experimentation in Software Engineering: An Introduction*. Springer-Verlag Berlin Heidelberg, 1st. edition, 2012.

[20] V. Basili and D. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738, 1984.

[21] José M. Chaves-González, Miguel A. Pérez-Toledano, and Amparo Navasa. Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. *Knowledge-Based Systems*, 83:105 – 115, 2015. doi: https://doi.org/10.1016/j.knosys.2015.03.012.

[22] Fitsum Meshesha Kifetew, Angelo Susi, Denisse Muñante, Anna Perini, Alberto Siena, and Paolo Busetta. Towards multi-decision-maker requirements prioritisation via multi-objective optimisation. In *CAiSE-Forum-DC*, pages 137–144, 2017.

[23] Aruan G Amaral and Glêdson Elias. A multi-objective, risk-based approach for selecting software requirements. In *10th International Conference on Agents and Artificial Intelligence*, pages 338–346, 2018.