Human Resource Optimization for Bug Fixing: Balancing Short-Term and Long-Term Objectives

Elias Khalil, Mustafa Assaf, and Abdel Salam Sayyad

Master Program in Software Engineering, Birzeit University
eliasdkh@gmail.com, mustsaf@gmal.com, asayyad@birzeit.edu

Abstract. In software development projects, bugs are usually accumulated and technical debt gets bigger over time. Managers decide to reduce the technical debt by planning one or more iterations for bug fixing. The time required to fix a bug depends on the required skill and the resource skill level. Managers seek to achieve fixing the highest number of bugs during the iteration while at the same time fixing the highest possible number of high severity and high priority bugs. In this study, we optimize the human resource assignment to achieve the objectives above, using multi-objective evolutionary algorithms, and then we add a fourth objective, i.e. that the bugs left out of the iteration should require the least time to finish. We show that the additional objective can be optimized without the detriment of other objectives. The lesson is that complicating the multi-objective problem formulation can help with the overall quality of the solutions.

Keywords: Human Resource Allocation, Software Project Planning, Agile Development, Search-Based Software Engineering.

1 Introduction

Software products should be delivered with high quality within time and budget. This requires spending high effort on testing and maintenance [11, 2]. Statistics showed that 80% of development cost is spent on bug fixing [10]. Unfortunately, project budget is usually limited, where just 32% of software projects are completed on time and within budget [4]. These facts raise the need to balance the effort spent on testing and maintenance and the limited schedule and budget.

Prior studies showed that selecting bugs to fix within a target planning period, and the criteria of assigning bugs to developers are significant factors affecting time to fix bugs [7, 1, 11]. Additionally, assigning a bug to the right human resource has a significant impact on the maintenance period or on the amount or nature of the fixed bugs during this period.

Resource allocation is considered an NP-Hard complex problem [3]. The complexity of this problem arises from the high number of combinations of possible allocation and the impact of the allocation on product development time, cost and quality, and overall project success.

This study focuses on human resource allocation for bugs within an agile process. Developing a product in an agile process passes through a sequence of iterations, each having a fixed period. By the end of the iteration, some bugs are moved to a backlog to be fixed in upcoming iterations. Hence, bugs accumulate in the backlog and the technical debt gets larger. At a certain point, management decides to plan one or more iterations dedicated to reducing the technical debit. The fact that resource allocation impacts the plan output raises the need to identify a close-to-optimal HR allocation using SBSE techniques.

The contribution of this paper is that it introduces and evaluates two formulations of the problem; one with three objectives (total bugs fixed, severe bugs fixed and high priority bugs fixed); and another where the time left to fix remaining bugs is added as a fourth objective. We show how adding the long-term quality objective does not hurt the quality of the three-objective case.

The rest of the paper is organized as follows: In section 2 we describe experiment design and dataset. In section 3 we show our results. And we provide a conclusion in section 4.

2 Experimental Setup

In this section, we discuss the setup of our experiment, including the optimization method, chromosome structure, fitness evaluation, experiment configuration, and the dataset.

2.1 Multi-Objective Optimization

Multi objective algorithms provide a Pareto front of nondominated solutions. A solution $x^{(1)}$ is said to be dominating $x^{(2)}$ if $x^{(1)}$ is not worse than $x^{(2)}$ in all objectives and $x^{(1)}$ is better than $x^{(2)}$ in one or more objectives [9].

Many search-based algorithms have been studied in the past 20 years. Evolutionary algorithms has provided significant solutions for both single and multi objectives search based problem. For this study, we use Non-dominated Sorting Genetic Algorithm II (NSGAII)[5], which is the most widely used algorithm in Pareto-Optimal SBSE [8].

2.2 Chromosome structure

A bug distribution plan is considered as an optimization solution where each bug is assigned to one developer. A list of bugs are assigned to a single developer. A binary solution is used to represent a bug distribution plan. It consists of a number of binary genes representing bugs to be fixed in the iteration as shown in Figure 1. Each Chromosome contains n bugs (genes) and each gene consists of m bits representing the developer id, in addition to 5 bits as a sequence number. This sequence number is used to handle the sequence that the developer will follow to fix bugs assigned to her.

Solutions usually have more than one bug assigned to a single developer. Thus two or more bugs may have the same sequence number. To solve this issue, a pseudo random number generator (PRNG) with this number as a seed is used to generate a list of unique numbers $gSeq_i$ where i is between 1 and number of bugs (n). Bug i with less $gSeq_i$ is planned to be fixed first.

Fig. 1. Chromosome Structure.

2.3 Multi-Objective Fitness Evaluation

Every bug has an estimated ETA set by a developer or manager. Usually this ETA is estimated based on an average skill level. A low skill level developer working on a medium or high skill level bug will spend more time to fix it. The opposite is also valid where a developer working on a bug requiring a skill level lower than the developer level, is expected to fix it in a time less than the bug ETA. Adjusted time for the developer to fix a bug ($Adjusted \quad ETA$) depends on both the bug ETA , developer skill level ($Skill_i$) for the specific bug category and bug required skill level (Bug_iSkill)

$$Skill_i \quad = \quad f(dev_{id}, \quad category(bug_{id}), \quad Bug_iSkill) \tag{1}$$

Adjusted estimated time to fix a bug i can be estimated by the skill level of the developer $skill_i$ relative to the bug category

$$Adjusted \quad ETA \quad = \quad ETA \quad f(\frac{skill_i}{required \quad skill}) \tag{2}$$

Each solution is a sequence of bugs to be fixed:

$$dev_i \quad List \quad = \quad \{B_{s_1}, B_{s_2}....B_{s_n}\} \tag{3}$$

A developer can fix number of bugs on the planned bugs list of the solution during iteration time T_{it}. This can be calculated by summing up the bugs fix time sequentially (Bs_i) until adding one more bug exceeds the iteration period.

$$\begin{aligned} BL_{it}(Dev_i) = \quad & \{B_{s_1}, B_{s_2} \quad .... \quad B_{s_t}\} \\ & where \quad \sum_{j=1}^{t} adjustedETA \quad (B_{s_j}) \quad \leq \quad IterationTime \end{aligned} \tag{4}$$

Based on iteration time and bugs assigned to each developer, Total bugs fixed in an iteration $B - Fixed_{it}$ is defined as union of bugs assign to each developer.

$$Fixed_{it} = \quad \cup_{i=1}^{d} \quad BL_{it}(Dev_i) \tag{5}$$

An additional objective outside of the iteration scope is added to the optimization. This objective represents the time required to finish all bugs in the backlog which should be minimized. This time can be calculated by summing the time required to fix each developer bugs.

$$TotalTime \quad = \sum_{i=1}^{n} adjustedETA \quad (B_i) \tag{6}$$

In summary, the objective fitness are represented using the following values:

1. Total number of bugs fixed in the iteration
2. High severity bugs fixed in the iteration
3. High priority bugs fixed in the iteration
4. Total period to fix all bugs including the iteration planned bugs

2.4 jMetal Study

jMetal framework [6] is used to build and run the study for both three and four objectives. Default jMetal configuration is used as tunning the algorithms for better results is not the purpose of this study.

Table 1. jMetal Experiment Configuration

Population size	100	Crossover type	Single Point
Crossover probability	0.9	Mutation type	Bit flip
Mutation probability	0.01	Max Evaluations	100,000

2.5 Dataset

The data provided for this study was extracted from a real bugs repository. Bugs were selected randomly for each set used in the study. For privacy reasons, the title, description, and other properties indicating any relation to the organization, industry or product were omitted. The remaining properties are the significant properties related to this study including ETA, severity, priority and required skill level. Additionally, a product category or module is added as a value between 1 to 5 representing 5 categories of the product where category name is removed for the same confidentiality reasons.

The trimmed dataset used by this study is hosted on the cloud[1]. It shows two employee files that include four or eight employees data. Skill level is ordered

[1] http://bzu.cloud:8080/BugsSBSE/resources/

from lowest to highest as 1,2,3. Additionally, each employee is rated with a skill 1 to 3 on 5 product categories. In addition, two sets of 50 or 100 bugs and their properties are included.

3 Results

A jMetal experiment setup is used for both three and four objectives. jMetal is configured for 40 runs on 4 problem setups as follows: 1)50 bugs, 4 developers. 2)100 bugs, 4 developers. 3)50 bugs, 8 developers. 4)100 bugs, 8 developers.

The above setup was executed twice. Once on the three objectives (excluding total time objective from the fitness) and another time for the four-objective case.

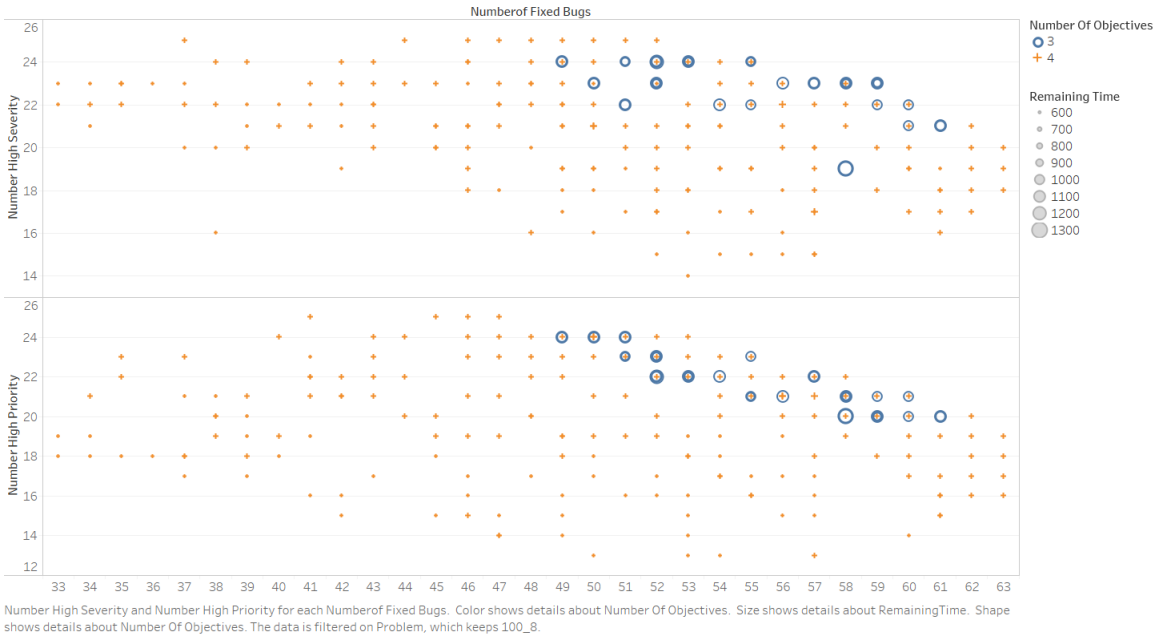

Fig. 2. Pareto Front of Three and Four Objectives Displayed in 2D

Figure 2 is used to present the three and four objectives as a relation between total number of fixed bugs and each of number of fixed severe bugs and number of fixed priority bugs separately. This chart presents the 100-bug 8-developer setup. Three-objective Pareto points are represented as crosses while the four-objective points are presented as circles. In order to add a time dimension to this 2D chart, the size of circle or cross points is used to represent the time required to fix all bugs.

Figure 2 shows that the circles size is mostly big which reflects the bad achievement of this objective. Additionally, it is clear that there are more points

in the four-objective Pareto front, providing more distribution of solutions while being able to achieve the same range achieved by the three objectives.

4 Conclusion

This study has adopted a search-based metaheuristic approach for human resource allocation for bug fixing iteration planning. Taking bugs severity, priority, ETA and required skill level in addition to developer skill level makes the choices more complicated for a manager planning such an iteration. We have showed that complicating the objectives by adding a long-term quality objective does not hurt the three objectives when compared with three-objective optimization results. In other words, complicating the problem formulation added quality to the recommended solutions.

References

[1] Anvik, J.: Automating bug report assignment. In: Proc. ICSE. (2006) 937–940

[2] Basili, V., Briand, L., Condon, S., Kim, Y.M., Melo, W.L., Valett, J.D.: Understanding and predicting the process of software maintenance release. In: Proc. ICSE. (1996) 464–474

[3] Bibi, N., Ahsan, A., Anwar, Z.: Project resource allocation optimization using search based software engineeringa framework. In: Proc. ICDIM. (2014) 226–229

[4] Chen, W.N., Zhang, J.: Ant colony optimization for software project scheduling and staffing with an event-based scheduler. IEEE TSE **39**(1) (2013) 1–17

[5] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE TEC **6**(2) (2002) 182–197

[6] Durillo, J.J., Nebro, A.J.: jmetal: A java framework for multi-objective optimization. Advances in Engineering Software **42**(10) (2011) 760–771

[7] Kang, D., Jung, J., Bae, D.H.: Constraint-based human resource allocation in software projects. S: Practice and Experience **41**(5) (2011) 551–577

[8] Sayyad, A.S., Ammar, H.: Pareto-optimal search-based software engineering (posbse): A literature survey. In: Proc. RAISE. (2013) 21–27

[9] Sayyad, A.S., Menzies, T., Ammar, H.: On the value of user preferences in search-based software engineering: a case study in software product lines. In: Proc. ICSE. (2013) 492–501

[10] Tassey, G.: The economic impacts of inadequate infrastructure for software testing. NIST, RTI Project **7007**(011) (2002)

[11] Zhang, F., Khomh, F., Zou, Y., Hassan, A.E.: An empirical study on factors impacting bug fixing time. In: Proc. WCRE. (2012) 225–234