# Towards Multi-decision-maker Requirements Prioritisation via Multi-Objective Optimisation

Fitsum Meshesha Kifetew[1], Angelo Susi[1], Denisse Muñante[1], Anna Perini[1], and Alberto Siena[2] and Paolo Busetta[2]

[1] Fondazione Bruno Kessler (FBK)
kifetew,susi,munante,perini@fbk.eu
[2] Delta Informatica
alberto.siena,paolo.busetta@deltainformatica.eu

**Abstract.** Requirements prioritisation is a key decision making activity of the software development process, which relies on the capability of different decision-makers to identify the optimal candidate rankings of the requirements, in order to be able to perform a strategic choice among them. In this paper, we formulate such *multi-decision-maker requirements prioritisation* as a *multi-objective optimisation problem*, and outline a solution that takes advantage of metaheuristic algorithms. The proposed solution allows different decision-makers to specify their rankings according to a set of prioritisation criteria, it then synthesises them into a set of Pareto optimal *global rankings*. The ultimate choice of the prioritisation of the requirements would be made upon a focused analysis of the trade-offs amongst the solutions on the Pareto frontier.

**Keywords:** Requirements prioritisation; multi-decision-maker prioritisation; evolutionary algorithms; multi-objective optimisation

## 1 Introduction

Requirements prioritisation is an important decision-making activity in the software development process that comes into the scene when the needs and desires of customers must meet the delivery capacity of the development team [1]. Once requirements have been successfully gathered, constraints and limitations create a trade-off between the opportunity gained from implementing the largest possible part of them, and the disadvantage of attempting to implement too many of them. In this case, knowing which are the best candidate requirements worthy of passing to the implementation phase allows to reduce the risk of failing to deliver the product.

Practitioners mostly carry out the requirements prioritisation activity with traditional approaches, such as focus groups. Several approaches have also been proposed to guide practitioners perform requirements prioritisation in a structured way [1]. A variety of automatic techniques to reduce the human effort in this process have been defined, which use, for example, the Analytical Hierarchy

X. Franch, J. Ralyté, R. Matulevičius, C. Salinesi, and R. Wieringa (Eds.):
CAiSE 2017 Forum and Doctoral Consortium Papers, pp. 137-144, 2017.

Process (AHP) [15, 2] method, and approaches based on Machine learning [11], or Constraint Satisfaction [10]. However, such activity can become a **complex decision-making problem** in the presence of certain conditions, e.g., (a) the number of candidate requirements increases, (b) the number of criteria for the prioritisation increase, (c) the dependencies between requirements are not negligible, or (d) different stakeholders are involved in the decision-making process, with their specific roles and competences. In this situation, the challenge becomes that of identifying the set of global rankings that better meet the various perspectives, thus ensuring the optimality of the final requirements prioritisation decision.

We observed examples of this situation in two small-medium companies in the context of the SUPERSEDE [3] project, a project focusing on feedback-driven adaptation and evolution of software systems. The first company is SEnerCon, a German company working in the domain of energy efficiency management, providing online applications in support of household energy saving. SEnerCon reports that when major innovation of their products need to be planned, different actors are relevant in prioritising the candidate requirements that might include the new ones that have been identified along the funding opportunities [9].

The second company is Delta, an Italian company working in serious gaming for professional training. As part of an industrial project, called PRESTO [13], Delta is working with researchers in the area of serious gaming and interaction design, and with domain experts to develop a virtual reality application for training in emergency management. Also in this case, actors with different roles and expertise contribute to prioritise the requirements to be considered for development.

In such situations, each actor involved in the prioritisation process has his/her own view regarding the priorities of the requirements, depending on their expertise and goals. Furthermore, these views of the various actors are typically of conflicting nature. For instance, developers may tend to give higher priorities to requirements that could be implemented with less effort, while experts from management may give higher priorities to requirements that would bring higher customer satisfaction (even if they incur higher development effort). Hence, the problem then becomes how to reconcile these individual views on the priorities into a final prioritisation of the requirements. We refer to the human actors involved in such a process, by providing their own preferences regarding the prioritisation of the requirements, as *decision-makers (DMs)*; and we refer to the problem as a *Multi-decision-maker Requirements Prioritisation* problem.

In this paper, we formulate the *Multi-decision-maker Requirements Prioritisation* problem as a multi-objective optimisation problem, and propose an approach that uses Evolutionary Algorithms (EA) to find optimal solutions to it. Given a set of rankings for the candidate requirements, which are made by different experts along multiple criteria, the proposed method allows to synthesise global requirements prioritisations located on a Pareto optimal frontier, thus

[3] www.supersede.eu

supporting the decision-maker by providing him/her with a view on alternative optimal requirements prioritisations.

The main contributions of this paper include: (*i*) a generic formulation of the *Multi-decision-maker Requirements Prioritisation* problem; (*ii*) an approach based on multi-objective optimisation for finding optimal solutions to this problem.

The rest of this paper is organised as follows: in Section 2, the formulation of the problem is given. The proposed approach for *Multi-decision-maker Requirements Prioritisation* is outlined in Section 3. The related works are discussed in 4. Finally, conclusions are presented in Section 5.

## 2 Problem Formulation

In a generic requirements prioritisation process, **multiple decision-makers** involved in a software project express independently their own priority list. Each decision-maker has a **relative weight** with respect to the other decision-makers, depending on aspects such as his/her role, expertise and so on. Moreover, any number of **evaluation criteria**, each with its own **weight** with respect to the expected final decision, can be used to perform the prioritisation. These criteria could represent positive (values) or negative (costs) evaluations about a given prioritisation of requirements, what is important here is that decision-makers are not obliged to summarise their positive (or negative) considerations along one single criterion. Furthermore, the requirements to be prioritised have their own characteristics, which must be taken into consideration when deciding on the final ranking, such as **dependencies** among requirements.

The ***Multi-decision-maker Requirements Prioritisation*** problem consists of finding the best ranking that takes into account all the individual rankings of the various decision-makers. Reaching consensus regarding the final ranking of the requirements out of the individual rankings, given by the decision-makers, involves: (i) identifying the alternative global rankings that can be drawn from the decision-makers' opinions; and (ii) selecting the final ranking, through a decision process that takes into account additional contextual and strategic information. While the final decision remains a strategic action, the *Multi-decision-maker Requirements Prioritisation* problem concerns supporting the identification of the optimal rankings.

In summary, given:

- a set of $n$ decision-makers who providing their prioritisations;
- a set of $m$ requirements to be prioritised (not necessarily independent);
- a set of dependencies where $(R_i \rightarrow R_j)$ implies that requirement $R_i$ depends on requirement $R_j$ ;
- a set of $k$ criteria along which the prioritisation is to be performed;
- weights corresponding to each criterion;
- weights corresponding to each decision-maker for each criterion;
- the prioritisations of each decision-maker with respect to each criterion;

the *Multi-decision-maker Requirements Prioritisation* problem consists of **finding the final prioritisation (ranking) of requirements with the minimum possible *distance* from the individual prioritisations (rankings) of *all decision-makers*, while respecting the constraints imposed by the *dependencies***. *Distance* represents a quantification of the **dissimilarity** between rankings.

## 3 Multi-Decision-Maker Prioritisation

The problem formulation presented in the previous section evidences the inherent multi-objective nature of the problem. In fact, the optimal solution to the problem involves trade-offs in different directions, in particular among the prioritisation criteria and among the preferences of the various decision-makers involved. Our proposed approach is based on the notion of minimising the *dissimilarity* among the preferences of the various decision-makers involved in the process. Given the preferences of each decision-maker with respect to each criterion, we try to find the prioritisation that is at the least possible level of dissimilarity from all the prioritisations of all decision-makers — the **middle ground**. In general, given a set of $m$ requirements, there are $m!$ possible prioritisation of these requirements. Hence, our approach employs multi-objective optimisation to explore this space of $m!$ candidate solutions with the objective of finding those with the minimum levels of *dissimilarity* from those of the decision-makers. Considering the potentially large number of alternatives, our proposed approach is based on Evolutionary Algorithms (EAs) [4] which are proven to be effective at scaling to large search spaces while at the same time providing optimal solutions for smaller spaces as well.

EAs are a class of metaheuristic search algorithms inspired by the process of natural evolution in which a *population* of candidate solutions (*individuals*) interact with each other and evolve through *generations* following the principle of survival of the fittest [4]. EAs are widely used to solve practical optimisation problems for which exact solutions could not be found in reasonable time. Furthermore, EAs follow a *global search* strategy which is quite robust in exploring the search space and finding globally optimal solutions by avoiding being trapped in *local optimal*, a phenomenon commonly associated to *local search* algorithms.

The right side of Figure 1 depicts a simplified overview of the main aspects of a typical EA, which starts by creating an initial population of individuals. It then evaluates each individual in the population by means of a *fitness function* and assigns it a *fitness value*. The EA then proceeds by selecting 'fitter' individuals (*parents*) from the current population and subjects them to the process of *reproduction* or *crossover* resulting in *offspring*. The offspring could further be subjected to a process of *mutation* with the aim of introducing diversity into the population. The EA then selects, from the combined pool of parents and offspring, the individuals that form the new population in the next generation (survivors). This process continues to iterate until some *stopping condition* is reached, in which case the EA terminates.

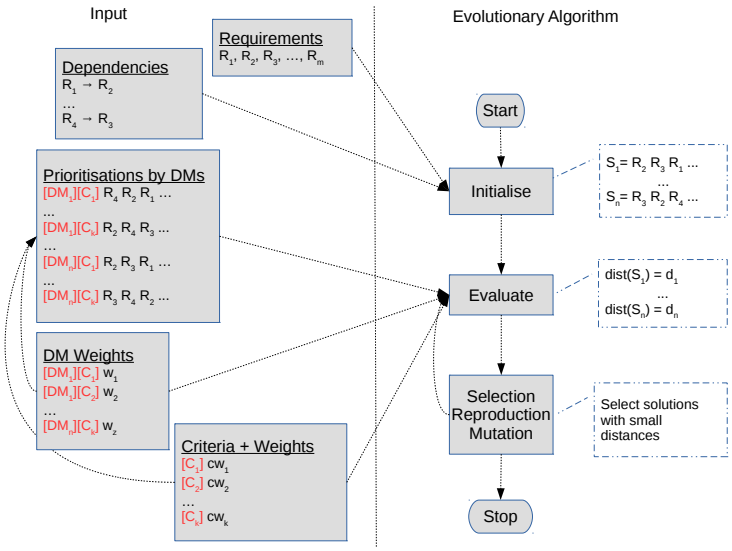
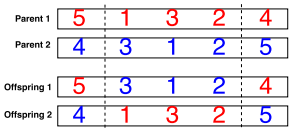



Fig. 1: Overview of solution. Elements in the problem formulation (Input) are mapped into the main phases of EA. *Requirements* and *Dependencies* are used to encode and initialise population, *Proritisations by DMs*, *Criteria*, and *Weights* are used to evaluate solutions using the fitness function.

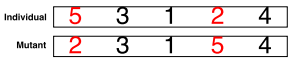
Given the formulation of the multi-decision-maker requirements prioritisation problem outlined in Section 2, an instance of an EA could be applied to search within the space of all possible prioritisations, with the ultimate goal of finding one or more prioritisation(s) with *minimal dissimilarity* from all of the given rankings. To this end, we need to appropriately define the corresponding EA operations in such a way that we are able to find optimal solutions to our problem. Specifically, we need to define (1) the encoding and initialisation of individuals, (2) the fitness function for evaluating individuals with respect to the problem, and (3) the selection, crossover, and mutation operators that allow the EA to evolve the individuals through generations.

**Solution encoding and initialisation:** Given $m$ requirements to be prioritised, candidate solutions (individuals) are prioritisations (rankings) of these requirements. Hence, we encode an individual as: $(r_1, r_2, \ldots, r_m)$ where $r_i$ represents the *rank* of *requirement i*. For example, if $m = 5$, an individual could be: (1,3,2,5,4). In this individual, the rank of *requirement 1* is 1, the rank of *requirement 2* is 3, the rank of *requirement 3* is 2, etc.

**Fitness function:** The goal of the fitness function is to objectively measure how good (fit) an individual is with respect to the problem being solved. For our problem, a good individual is one that has the lowest level of dissimilarity (disagreement) among the prioritisations of the decision-makers. Hence, we first quantify the level of dissimilarity between two rankings by means of a *distance*




(a) Two Point Crossover




(b) Swap Mutation

Fig. 2: EA Operators

function. Given two individuals $I_1$ and $I_2$ representing two rankings of $m$ requirements, the distance between them could be computed in a number of ways. For instance, we can define a distance metric based on the (average) differences between the ranks of the requirements:

$$d(I_1, I_2) = \sum_{i=1}^{m} |I_1[i] - I_2[i]| \tag{1}$$

Similarly, other rank similarity metrics, such as *Kendall's* $\tau$ statistic [7], could be used to define a distance function:

$$d(I_1, I_2) = 1 - KendallTau(I_1, I_2) \tag{2}$$

Based on the distance functions given in Equations 1 and 2, we define *fitness functions* for computing the fitness of an individual $I$ in the EA. Specifically, we consider a *multi-objective* perspective in which we compute the *distance* of $I$ with respect to more than one objective. For our problem formulation, we define a fitness function considering the distance of $I$ from each *criterion*.

The EA searches for individual(s) with the minimum values for the fitness functions, *i.e.,* solutions that represent minimal dissimilarity among the rankings of the various decision-makers.

**EA operators:** Here we describe the two most important operators, *crossover* and *mutation*, that enable the EA to evolve individuals through generations.
***Two point crossover*** in each individual's (*parent*) encoding, two points are randomly picked, and parts of the individual's encoding are exchanged. Figure 2a illustrates this crossover operation. In case the crossover operator results in invalid individuals (*e.g.,* duplicated ranks), the individual will either be corrected in a subsequent phase, or the operation will be cancelled and performed again choosing different points of crossover.
***Swap mutation*** two points are randomly picked in the individual's encoding, and the values indicated by the selected points (indices) are *swapped*. Figure 2b illustrates this mutation operation.

**Algorithm:** Given the problem formulation, the encoding of individuals, and the operators defined above, our approach employs a *multi-objective* EA (*e.g.,* NSGA-II [3]) based on the multi-objective fitness function described above to find a set of *Pareto optimal* solutions. Each solution in the Pareto frontier represents a trade-off within the space of the objectives being optimised. Hence, depending on the currently sought solution and context, the human expert should choose one of the solutions.

**Dependencies** among requirements could be handled in two ways:
*As hard constraints:* candidate solutions that violate the dependency constraints will not be considered as valid candidates and hence will be discarded.
*As soft constraints:* dependency constraint violation is computed as a *secondary objective* to be considered in case of equivalence in the first objective (*i.e.,* dissimilarity). Candidate solutions that violate fewer constraints will be preferred in case of a tie between candidates with respect to their distances. This approach is useful in cases where the number of dependencies is high, and consequently the number of valid solutions is low.

In summary, the approach will search for optimal prioritisations that minimise dissimilarity (disagreement) among the various DMs involved in process, finding the *middle ground*.

## 4 Related Works

Requirements prioritisation is an area of requirements engineering which has received a significant amount of attention both from the research community and industry [1]. This is mainly attributed to the fact that the decisions taken during prioritisation could have profound effects on strategic as well as technical (operational) aspects of an organisation [14, 8]. Relevant work identified and discussed the challenges of multi-stakeholder prioritisation from the perspective of traditional (closed) organisation, e.g. [16, 12], and mostly addressed the multi-decision-maker issue with negotiation approaches. Concerning the automatic techniques used in requirements prioritisation, several approaches have been presented that use constraints based techniques and search-based techniques such as Satisfiability Modulo Theory techniques, and heuristic based techniques, in particular genetic algorithms [10, 17]. Considering the latter techniques several search based approaches have been exploited for the solution of different kinds of requirements engineering problems such as multi-objective requirements prioritisation and next release problem [6] also in presence of multiple customers [5]. Our approach considers some of the issues and observations reported in the mentioned works and aims at extending them to offer a distributed mechanism for eliciting preferences from stakeholders with potentially different skills and expertise, by employing a collaborative process that allow to find optimal trade-offs among those preferences.

## 5 Conclusion

We presented a multi-objective formulation of the multi-decision-maker requirements prioritisation problem, and outlined a solution based on Evolutionary Algorithms. The proposed approach is based on the notion of finding Pareto optimal prioritisations that exhibit the minimum levels of disagreement among the various decision-makers involved in the process. The ultimate decision-maker selects one of the optimal solutions based on additional interests (*e.g.,* strategic) not necessarily included in the prioritisation criteria. The work is currently

ongoing and we are working towards an empirical evaluation of the proposed approach on real world case studies derived from the industry.

**Acknowledgement.** This work is a result of the SUPERSEDE project, funded by the H2020 EU Framework Programme under agreement number 644018.

## References

1. P. Berander and A. Andrews. Requirements Prioritization. In A. Aurum and C. Wohlin, editors, *Engineering and Managing Soft. Requirements*. Springer, 2005.
2. P. Busetta, F. M. Kifetew, D. Munante, A. Perini, A. Siena, and A. Susi. Tool-supported Collaborative Requirements Prioritisation. In *Proceedings of the IEEE Int. Conference COMPSAC, Torino, Italy, July 4-7*, 2017. To appear.
3. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. on Evol. Comp.*, 6:182–197, 2000.
4. A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Springer Science & Business Media, 2003.
5. A. Finkelstein, M. Harman, S. Mansouri, J. Ren, and Y. Zhang. A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. *Requirements engineering*, 14(4):231–245, 2009.
6. D. Greer and G. Ruhe. Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, 46(4):243–253, 2004.
7. M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.
8. N. B. Moe, A. Aurum, and T. Dybå. Challenges of shared decision-making: A multiple case study of agile software development. *Inf. Softw. Technol.*, 54(8):853–865, Aug. 2012.
9. D. Muñante, F. M. Kifetew, and O. Albrecht. Modelling prioritisation decision-making in software evolution. In *Joint Proceedings of REFSQ-2017 Workshops, Doctoral Symposium, Research Method Track, and Poster Track co-located with the 22nd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2017), Essen, Germany, February 27, 2017.*, 2017.
10. F. Palma, A. Susi, and P. Tonella. Using an smt solver for interactive requirements prioritization. ESEC/FSE '11, pages 48–58, New York, NY, USA, 2011. ACM.
11. A. Perini, A. Susi, and P. Avesani. A Machine Learning Approach to Software Requirements Prioritization. *Software Engineering, IEEE Transactions on*, 39(4):445–461, 2013.
12. B. Regnell, M. Höst, J. N. och Dag, P. Beremark, and T. Hjelm. An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. *Requirements Engineering*, 6(1):51–62, 2001.
13. M. Robol and P. Busetta. Applying bdi to serious games: The presto experience. Technical report, Universita di Trento, 2016.
14. G. Ruhe and M. O. Saliu. The art and science of software release planning. *IEEE Softw.*, 22(6):47–53, Nov. 2005.
15. T. L. Saaty. What is the analytic hierarchy process? In *Mathematical models for decision support*, pages 109–121. Springer, 1988.
16. V. Sinha, B. Sengupta, and S. Chandra. Enabling collaboration in distributed requirements management. *Software, IEEE*, 23(5):52–61, 2006.
17. P. Tonella, A. Susi, and F. Palma. Interactive requirements prioritization using a genetic algorithm. *Inf. Softw. Technol.*, 55(1):173–187, Jan. 2013.