This is the Author's Accepted Manuscript of an article published in REFSQ 2021: Requirements Engineering: Foundation for Software Quality. The final version is available via Springer at: https://doi.org/10.1007/978-3-030-73128-1_8

Defining Utility Functions for Multi-Stakeholder Self-Adaptive Systems

Rebekka Wohlrab✉[0000−0002−5449−7900] and David Garlan[0000−0002−6735−8301]

School of Computer Science, Carnegie Mellon University, USA
wohlrab@cmu.edu, garlan@cs.cmu.edu

arXiv:2103.10101v1 [cs.SE] 18 Mar 2021

Abstract. **[Context and motivation:]** For realistic self-adaptive systems, multiple quality attributes need to be considered and traded off against each other. These quality attributes are commonly encoded in a *utility function*, for instance, a weighted sum of relevant objectives. **[Question/problem:]** The research agenda for requirements engineering for self-adaptive systems has raised the need for decision-making techniques that consider the trade-offs and priorities of multiple objectives. Human stakeholders need to be engaged in the decision-making process so that the relative importance of each objective can be correctly elicited. **[Principal ideas/results:]** This research preview paper presents a method that supports multiple stakeholders in prioritizing relevant quality attributes, negotiating priorities to reach an agreement, and giving input to define utility functions for self-adaptive systems. **[Contribution:]** The proposed method constitutes a lightweight solution for utility function definition. It can be applied by practitioners and researchers who aim to develop self-adaptive systems that meet stakeholders' requirements. We present details of our plan to study the application of our method using a case study.

Keywords: self-adaptive systems · quality attributes · utility functions · Analytic Hierarchy Process.

1 Introduction

For self-adaptive systems, multiple quality attributes (such as performance, availability, and security) need to be considered and traded off against each other. These quality attributes are often encoded in a *utility function*, i.e., a single aggregate function whose expected value should be maximized by the system [4, 6, 8, 16]. In self-adaptive systems, utility functions are typically used by automated planning mechanisms to identify the relative costs and benefits of alternative strategies. In related work, utility functions are often defined as the weighted sum of relevant objectives [3, 5, 18]. For most approaches using utility functions, it is simply stated that they should be manually defined, but little guidance for this task is provided [8, 18]. It is challenging to correctly identify the weights of each objective and consider trade-offs between multiple quality attributes, as reported in the research agenda for requirements engineering for self-adaptive systems [16]. Self-adaptive systems often have multiple stakeholders

(e.g., end users or business owners) whose preferences need to be consolidated to identify the overall relative importance of each objective [15]. Decision-making techniques are needed to help stakeholders prioritize and negotiate quality attributes and determine appropriate utility function weights [16].

In this paper, we present a lightweight tool-supported method for utility function definition for multi-stakeholder self-adaptive systems. The proposed method is based on the Analytic Hierarchy Process (AHP) [14] and the Delphi technique [7]. It supports stakeholders in prioritizing quality attributes, negotiating priorities to reach an agreement, recording rationales and comments, and giving input to define utility functions. We use the weighted sum approach for utility functions, as it is lightweight and commonly used in related work [3, 5, 18]. It assumes that the weighted quantity of one quality attribute can be traded off (or "substituted" [1]) with another one. Guidelines to select utility functions, considering risk and dependencies between decision parameters, have been previously created [1] and can be used to support other kinds of utility functions in the future. The proposed ideas of this research preview paper will be further refined and we plan to study the method's application in a case study. We expect that our method will be of use to practitioners and researchers that aim to conceive self-adaptive systems fulfilling stakeholders' preferences.

2 Proposed Approach

Figure 1 shows the steps of our method for utility function definition. The method can either be used for the initial definition or the refinement of the utility function, in case stakeholders' preferences evolve over time. The two leftmost steps are performed individually by each stakeholder. The two guard conditions refer to whether an AHP matrix is inconsistent and whether no agreement has been reached. Each step is labeled with the paragraph in which it is described.

(A) Create an AHP Matrix: For the prioritization of quality attributes, we use the AHP, which is especially useful when subjective, abstract, or non-quantifiable criteria are relevant for a decision [14]. A central part of the AHP is to elicit stakeholders' priorities of different objectives in pairwise comparison matrices, which are positive and reciprocal (i.e., $a_{ij} = 1/a_{ji}$). For utility functions, we are interested in the degree of preference of one quality attribute over another, with the goal of increasing the overall utility of a system. Verbal expressions are used for these pairwise comparisons (e.g., "*I strongly prefer X over Y*"). Table 1 shows how the verbal expressions correspond to numerical values.

For a robot planning problem, Table 2 shows an example of an AHP matrix with the attributes safety (expected number of collisions), speed (duration of

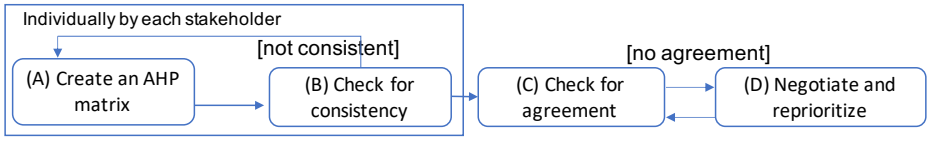


Fig. 1. Overview of our method for utility function definition

a mission), and energy consumption (consumed watt-hours). In the example, safety is *very strongly* preferred over speed (7) and *extremely preferred* over energy consumption (9). Speed and energy consumption are *equally preferred*.

The relative priorities of the quality attributes can then be calculated using the principal eigenvector of the eigenvalue problem $Aw = \lambda_{max} w$ [14]. A is the matrix of judgments and λ_{max} is the principal eigenvalue. For the matrix in Table 2, the principal eigenvalue is $\lambda_{max} \approx 3.01$. A corresponding normalized eigenvector to λ_{max} is $(0.8, 0.1, 0.1)^T$, which corresponds to the relative priorities of the quality attributes. The utility function for a mission might be defined as $U(m) = 0.8 \cdot \mathsf{safety}(m) + 0.1 \cdot \mathsf{duration}(m) + 0.1 \cdot \mathsf{energy}(m)$. $\mathsf{safety}(m)$ indicates the expected number of collisions in a mission, $\mathsf{duration}(m)$ the number of timesteps, and $\mathsf{energy}(m)$ the consumed watt-hours. The preference of a quality attribute can often be described with a sigmoid function defining an interval for the quantity that is considered as good enough and an interval for the quantity that is insufficient [12]. Appropriate methods will need to be created in the future to elicit these thresholds and define quality attributes' preference functions.

(B) Check for Consistency: AHP matrices can be checked for consistency. A matrix is consistent if $a_{jk} = a_{ik}/a_{ij}$ for $i, j, k = 1, \ldots, n$ [14]. Saaty proved that a necessary and sufficient condition for consistency is that the principal eigenvalue of A be equal to n, the order of A [14]. He defined the consistency index CI as $(\lambda_{max} - n)/(n-1)$. For our example in Section 2, CI is 0.004. To compare consistency values, Saaty also calculated the *random consistency index* RI by calculating CI for a large number of reciprocal matrices with random entries [14]. For a 3×3 matrix, the average random consistency index was 0.58. According to Saaty, the consistency ratio $CR = CI/RI$ shall be less or equal to 0.10 for the matrix to be considered consistent [14]. In our example, the consistency ratio is 0.01. If consistency is not fulfilled, stakeholders are required to refine their AHP matrices. The matrix can be automatically analyzed to point out the triples of quality attributes QA_i, QA_j, and QA_k where $a_{jk} \ll a_{ik}/a_{ij}$ or $a_{jk} \gg a_{ik}/a_{ij}$.

(C) Check for Agreement: We consider the rankings of n quality attributes by k stakeholders (where each quality attribute's rank is a number between 1 and n). For QA_i, the sum of ranks by all stakeholders is R_i, and the mean value of these ranks is $\bar{R} = \frac{1}{n}\sum_{i=1}^{n} R_i$. If the stakeholders' rankings do not agree, we can assume that the sums of ranks of several quality attributes are approximately equal [10]. It is therefore natural to consider the sum of squared deviations from the mean values of ranks $S = \sum_{i=1}^{n}(R_i - \bar{R})^2$ [10]. The maximum possible

Table 1. AHP judgment/preference options with numerical values [14].

Extremely preferred	9
Very strongly preferred	7
Strongly preferred	5
Moderately preferred	3
Equally preferred	1
Intermediate values	2, 4, 6, 8

Table 2. Example of an AHP matrix.

	Safety	Speed	Energy Consumption
Safety	1	7	9
Speed	$\frac{1}{7}$	1	1
Energy Cons.	$\frac{1}{9}$	1	1

value of S is $k^2(n^3-n)/12$ [10]. Kendall's concordance coefficient, describing the agreement of rankings in a [0,1] interval, is therefore: $W = \frac{12S}{k^2 \cdot (n^3-n)}$ [10].

(D) Negotiate and Reprioritize: In case agreement is not reached, a tool-supported negotiation and reprioritization phase starts. To aggregate AHP matrices, the "most recommendable aggregation technique" is to calculate the weighted arithmetic mean of individual priorities (AIP) [11]. A priority indicates the importance of a quality attribute with a value between 0 and 1. Stakeholders' priorities can be weighted differently, as their influence and stake may differ.

While the AIP can be used to quickly arrive at a solution, it is beneficial to discuss and record underlying rationales. We adapt the *Delphi technique* [7] for remote consensus building. Interactive tooling is used to support the technique and collect data. The stakeholders anonymously provide input in several iterations and receive controlled feedback. Users can declare that they do not know or do not care about a quality attribute. It is also possible to delegate votes to another participant (proxy voting). In the first round, open-ended questions are used concerning participants' rationales (e.g., "in what situation(s) do you think safety is especially important?"). The answer is fed back to other participants to inform their rankings. The main trade-offs and conflicts between quality attributes are elicited and discussed. While we assume an existing set of quality attributes, participants can also suggest new quality attributes and objectives.

In the second round, the comments and rationales are presented to the participants and the AHP matrices can be revised. The rankings that are in conflict are indicated to increase transparency. Further comments and rationales are added and a consensus starts to form. In the third round, participants are asked to revise their judgments or declare why they decide to remain outside the consensus [7]. The final utility function is a weighted sum of the objectives, where the final weights are the participants' aggregated weighted priorities (using AIP).

3 Empirical Study

We plan to perform a multiple case study [13] focusing on the phenomenon of applying our approach in practice. The approach depends on contextual factors and is therefore difficult to study in controlled settings (e.g., experiments). As mentioned before, utility functions are a central mechanism in several approaches for self-adaptive systems. We plan to apply the method to existing systems and projects. As the first case, we focus on robot mission planning using a probabilistic model checker, where the correct definition of the weights of multiple objectives is essential. The participants operate from multiple locations and are aware of the system's context and a preliminary set of quality attributes. The stakeholders have conflicting objectives (e.g., end user, business/cost, performance, and safety concerns) and are asked to apply our method for utility definition. We aim to use collected tool data, observations, and complementary interviews to study the required time to build a consensus, the understandability of the approach, as well as negotiation strategies. The empirical study is intended to give insights into how participants typically act to reach a consensus, how beneficial our proposed method is perceived for utility function definition, and how satisfied stakeholders are with the resulting utility function.

4 Related Work

Identifying and prioritizing objectives for self-adaptive systems is a nontrivial task. The Goal-Action-Attribute Model requires the goals, priorities, and preferences of multiple stakeholders to be elicited [15]. The AHP is suggested to be used for this task, but no concrete guidance is given. We focus on prioritization and negotiation and develop a comprehensive AHP-based method. Rather than focusing on creating complete goal models, we aim to create a lightweight method for utility function definition. For security requirements, the Swing-Weight Method has been used for prioritization and utility function definition [2]. The AHP allows a more precise elicitation of the relative priorities of objectives.

Utility functions are a common mechanism in self-adaptive systems [3–6, 18]. A few approaches for utility function definition are related to our work on the prioritization and negotiation of utility function weights. Song et al. [17] propose to collect user feedback after every round of adaptation to adjust the weights of constraints. Another approach relies on user feedback to switch between "variants" with associated utility function weights, depending on the current usage context [9]. Our work focuses on eliciting priorities to define utility function weights based on a consensus between multiple stakeholders. As part of future work, we aim to also consider different usage contexts/scenarios in our method.

5 Discussion, Conclusion, and Future Work

In this paper, we presented a method to define utility functions for self-adaptive systems by eliciting and negotiating the priorities of quality attributes. The method is based on the AHP for the pairwise comparison of quality attributes and a consensus-building approach using the Delphi technique. We plan to study the method's application on existing systems in a multiple case study. The method is at an early stage of investigation and needs to be refined further. For instance, the current method assumes that stakeholders are aware of relevant and measurable quality attributes that can be expressed in functions. For individual quality attributes, techniques are needed to define the intervals of values that are considered "good enough" or "insufficient." Our method will also be extended to explicitly consider hard constraints. Criteria mandated by law (e.g., security or safety constraints) cannot be traded against other preferences. Moreover, the utility of a system strongly depends on its context (e.g., current tasks, security attacks, or faults), which should also be considered, so that human input for utility function definition can be collected when needed and the utility function can be evolved over time. Another relevant concern is to ensure that stakeholders do not over-rate quality attributes to counter for others' conflicting preferences, as it is not always possible to assume non-competitive stakeholders.

We envision our method to be integrated into existing approaches, so that multiple stakeholders' preferences and requirements can be more easily elicited, negotiated, and fulfilled. The presented ideas might also be beneficial for artifacts at other levels of abstraction, e.g., the prioritization of goals or requirements. Moreover, we plan to work on explaining utility functions by describing different priorities' impact on the concrete actions of a system. Our vision is to demystify

utility functions by providing human stakeholders with lightweight and understandable methods for the definition and refinement of utility functions.

Acknowledgments: This work is supported in part by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by award N00014172899 from the Office of Naval Research and by the NSA under Award No. H9823018D000. Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research or the NSA.

References

1. Abdennadher, I., Rodriguez, I.B., Jmaiel, M.: A utility-based approach for self-adaptive systems: Application to a smart building. In: AICCSA. pp. 76–82 (2018)
2. Butler, S.A., Fischbeck, P.: Multi-attribute risk assessment. In: SREIS'02 (2002)
3. Cheng, S.W., Garlan, D., Schmerl, B.: Architecture-based self-adaptation in the presence of multiple objectives. In: SEAMS'06 (2006)
4. Faniyi, F., Lewis, P.R., et al.: Architecting self-aware software systems. In: WICSA'14. pp. 91–94 (2014)
5. Ghezzi, C., Molzam Sharifloo, A.: Dealing with Non-Functional Requirements for Adaptive Systems via Dynamic Software Product-Lines, pp. 191–213. Springer Berlin Heidelberg (2013)
6. Heaven, W., Sykes, D., et al.: A case study in goal-driven architectural adaptation. In: Software Engineering for Self-Adaptive Systems, pp. 109–127 (2009)
7. Hsu, C.C., Sandford, B.A.: The Delphi technique: making sense of consensus. Practical Assessment, Research, and Evaluation **12**(1), 10 (2007)
8. Inverardi, P., Mori, M.: A software lifecycle process to support consistent evolutions. In: de Lemos, R. (ed.) Self-Adaptive Systems, vol. 7475 LNCS, pp. 239–264. Springer Berlin Heidelberg (2013)
9. Kakousis, K., Paspallis, N., Papadopoulos, G.: Optimizing the utility function-based self-adaptive behavior of context-aware systems using user feedback. In: OTM 2008. pp. 657–674 (11 2008)
10. Kendall, M.G., Smith, B.B.: The problem of m rankings. Ann. Math. Statist. **10**(3), 275–287 (09 1939)
11. Ossadnik, W., Schinke, S., Kaspar, R.H.: Group aggregation techniques for analytic hierarchy process and analytic network process: A comparative analysis. Group Decision and Negotiation **25**(2), 421–457 (2016)
12. Poladian, V., Sousa, J.P., Garlan, D., Shaw, M.: Dynamic configuration of resource-aware services. In: ICSE 2004 (2004)
13. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering **14**(2), 131–164 (Apr 2009)
14. Saaty, R.: The analytic hierarchy process—what it is and how it is used. Mathematical Modelling **9**(3), 161–176 (1987)
15. Salehie, M., Tahvildari, L.: Towards a goal-driven approach to action selection in self-adaptive software. Software: Practice and Experience **42**(2), 211–233 (2012)
16. Sawyer, P., Bencomo, N., et al.: Requirements-aware systems: A research agenda for RE for self-adaptive systems. In: RE'10. pp. 95–103 (2010)
17. Song, H., Barrett, S., Clarke, A., Clarke, S.: Self-adaptation with end-user preferences: Using run-time models and constraint solving. In: MODELS'13 (2013)
18. Sousa, J.P., Balan, R.K., Poladian, V., Garlan, D., Satyanarayanan, M.: User guidance of resource-adaptive systems. In: ICSOFT 2008. pp. 36–44 (2008)