A Survey on Preferences of Quality Attributes in the Decision-making for Self-Adaptive Systems: the Bad, the Good and the Ugly

Luis H. Garcia Paucar and Nelly Bencomo

ALICE, Aston University, UK
garciapl@aston.ac.uk, Nelly@acm.org

Abstract. Different techniques have been used to specify preferences for quality attributes and decision-making strategies of self-adaptive systems (SAS). These preferences are defined during requirement specification and design time. Further, it is well known that correctly identifying the preferences associated with the quality attributes is a major difficulty. This is exacerbated in the case of SAS, as the preferences defined at design time may not apply to contexts found at runtime. This paper aims at making an exploration of the research landscape that have addressed decision-making and quality attribute preferences specification for self-adaptation, in order to identify new techniques that can improve the current state-of-the-art of decision-making to support self-adaptation. In this paper we (1) review different techniques that support decision-making for self-adaptation and identify limitations with respect to the identification of preferences and weights (i.e. the research gap), (2) identify existing solutions that deal with current limitations.

Keywords: Self-adaptation, decision making, preference trade-off, quality attributes

1 Introduction

Decision-making requires the quantification and trade-off of multiple quality attributes and the analysis of the costs and benefits between alternative solutions [1, 2]. Decision-making is at the core of self-adaptation [3]. An important issue is the specification of preferences associated with quality attributes and defined in the decision-making strategies to support self-adaptation [4]. Priorities associated with requirements may vary from stakeholder to stakeholder and from one envisaged situation to another. Different priorities can imply different decisions. Further, the assumptions made at requirements specifications and design time can change during runtime [5] and those changes can create alterations of priorities and utility preferences. This is exacerbated in the case of SAS, as the preferences defined during the requirements specification and design may not apply to contexts found at runtime and therefore causing unexpected or incorrect behaviour. Modeling and reasoning with prioritization and preferences is a research field that needs more attention [1]. Different studies of the state-of-the-art for SAS in general have been performed [6–8, 2, 9]. They have identified current approaches and trends but also critical challenges that must be further

explored. However, most of the current approaches tend to focus on design time issues and are effective in specific domains but unlikely to be widespread. Very few of these approaches address dynamic decision-making hence, it is still an issue [2]. Further, the need for updating utility preferences or uncovering relationships between non-functional requirements (NFRs) during runtime has been neglected. The steps of monitoring the environment, detecting the need for (self-) adaptation and deciding how to react are challenges for SAS [6], which still need to involve as well the role of preferences and re-prioritization of quality attributes during runtime. Our main research goal is to study the current approaches that tackle utility preference elicitation and prioritization with respect to quality attributes and also, investigate how these are updated during runtime when new relationships between quality attributes are found. In this study, we start from two main research questions that drive the work: (RQ1) *What are the current approaches that address preferences definition in SAS?* and (RQ2) *What are the current approaches that address dynamic re-assessment of preferences in SAS at runtime?* The aim of these questions is to identify the existing studies that explicitly consider the need to reassess utility preferences at runtime, to assess the current research results to identify research gaps. In order to answer these questions, we have compared a set of approaches to determine their main characteristics, benefits and limitations. We have used the results of different recent surveys in the area of decision-making for self-adaptive systems [6–8, 2, 9]. We have defined the classification criteria (in section 4), which allowed us to summarize the responses into two kinds of research findings: (i) those related to goal and decision making and (ii) those related to preferences. The review emphasizes the analysis on the second group (ii). However, such analysis includes references and the context provided by (i). Different techniques have been analyzed for the specification and update of preferences, to therefore determine the most appropriate approaches for self-adaptation.

The review is organized as follows. Section 2 describes the search strategy implemented to guarantee a systematic search approach. Section 3 shows an overview of the selected approaches. Section 4 presents an analysis of the approaches reviewed, identifying their generic characteristics, benefits, and limitations. Finally, Section 5 concludes the review and overviews ideas to fill the identified research gaps.

2 Search strategy

The selection criterion to increase the possibilities of having relevant studies was based on a search strategy. The search strategy involved papers published in English, in conferences and journals as full research papers, short and position papers about requirements trade-off and quality attribute preferences specification for SAS and based on recent surveys of the research area [6, 10, 8, 2, 9]. The digitial libraries selected for the search process are summarized in Table 1.

Specialized venues has been also identified in the field of Requirement Engineering and self-adaptive software. The targetted journals were:

- ACM Transactions on Autonomous and Adaptive Systems
- IEEE Transactions on Software Engineering

Table 1. Target Databases

Digital library/database	Url
IEEE	http://ieeexplore.ieee.org/
ACM Digital Library	http://dl.acm.org/
Scopus	https://www.scopus.com/
Web of Science	http://www.webofknowledge.com/
Springer	http://link.springer.com/

- Journal of Software and Systems (Elsevier)
- Information and Software Technology (Elsevier)
- Software and System Modelling (Springer)

For the specialized conferences, there have been identified:

- Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)
- Conference on Software Engineering (ICSE)
- Conference on Model-Driven Engineering Languages and Systems (MODELS)
- Requirement Engineering Conference (RE)

2.1 Search String

The aim of the search string is to capture all results related to non- functional requirements and preferences in the context of self-adaptive software. Several trial searches were performed in each database with the intention of checking the number of returned papers and their relevance. The objective of the trial searches is to check the feasibility of the search string and adjust it accordingly. The general search string used on all databases is: (preference*) AND (self*) AND (software). By performing the search, different results were obtained from the target databases: ACM (323 results), IEEE (56 results), SCOPUS (336 results), Web of Science (205 results) and Springer (18787 results). After applying the selection and exclusion criteria (See section 2.2), 10 papers were finally selected. The search was executed on the databases using the search string, as specified earlier. Minimizing results by excluding irrelevant disciplines was used, whenever available. In case the search engine does not imply enough filters and large number of irrelevant results were retrieved, there were used the first 100 search results ordered according to the relevance with regard to the search string. This decision was made after checking up another 100 search results after the first ones and found irrelevance.

2.2 Selection and exclusion criteria

Inclusion criteria has been defined with the aim of increasing the possibilities of having relevant studies. These are:

- Papers published in conferences and journals, as full research paper, short and position paper presenting new and emerging ideas.
- Papers presenting a technique for NFR preference definition or updating.
- Papers discussing aspects of NFR preferences definition or updating.

Exclusion criteria: Applied on the results retrieved, are:

- Duplicate studies.
- Papers in the form of abstract, tutorials, posters or presentation.
- Abstract not available.
- Full-text papers not accessible.
- Papers not explicitly addressing preference definition or updating.

3 Approaches overview

When we make decisions, a natural approach is to evaluate our different alternatives and choose one based on some criteria [11]. In SAS we must be able to apply this natural way of reasoning but with the challenge of working under uncertain conditions. How to ensure a reliable and accuracy behaviour (i.e., optimize the systems behaviour) trading-off multiple goals competing among them and being constantly affected by external changing conditions is the field of action of the well known set of methods called Multi Criteria Decision Analysis Methods (MCDA). MCDA methods are currently applied in different fields [12] but more than ever in self-adaptation. Different MCDA techniques have been used for both, decision-making and preferences specification in SAS. Some popular MCDA techniques such as Pareto Optimal [13] have been use for reasoning at runtime to discover a set of optimal adaptation alternatives. The final alternative selection could either demand the user intervention or be part of a fully autonomic system behaviour. Other MCDA approaches such as Analytic Hierarchical Process (AHP) [14] have also been used for specifiying quality attribute preferences at design time collected from system's stakeholders following a more formal and objective approach.

In the rest of this section we explore different approaches for decision-making in SAS, focusing on elicitation and dynamic update of preferences. The approaches covered are summarized in Table 2.

Table 2. Approaches for specifying requirements and preferences

Approach ID	Main author
AP1	Song et. al. (MODELS_2013)
AP2	Letier et al. (ICSE_2014)
AP3	García-Galán et al. 2014 (SEAMS_2014)
AP4	Elahi et. al. (SAC_2011)
AP5	Liaskos et al. (RE_2011)
AP6	Peng et al. (Journal of Systems and Software_2012)
AP7	Bencomo et al. (ICSE_2014)
AP8	Sousa et al.(ICSOFT_2008)
AP9	Ramirez et al.(MODELS_2011)
AP10	Walsh et al. (ICAC_2004)
AP11	Angelopoulos et al. (SEAMS 2014)

Song et.al. (AP1) Song et.al. [15] presents an approach where adaptation goals and structural runtime models (i.e. the current system context and configuration) are transformed into a Constraint Satisfaction Problem (CSP) [13] to simplify the model and facilitate reasoning. Within the CSP, the constraints are classified as hard (configuration domain values and current context values) and weak (configuration values and adaptation goals). Each weak constraint has a

weight (i.e. utility preference) based on the users preferences while the system quality attributes are part of the adaptation goals. Based on the CSP, the adaptation is performed in two stages: Constraint diagnosis (to determine the set of constraints to be ignored) and Constraint solving: to assign new configuration values that satisfy the remaining constraints. The algorithm proposed uses constraints' weights and is based on a dynamic programming approach. Preferences are elicited initially from information provided by system stakeholders and the model supports upgrade of preferences through an interactive process in which the user must participate. If users do not agree with the final solution (after each round of adaptation), they can therefore revise the configuration values using an interface. The proposal was applied to a case study based on a smart house however, the model could be applied in different domains.

Letier et.al. (AP2) Letier et.al. [16] proposed the representation of system goals (i.e. optimisation goals) as part of an Architecture Decision Model (ADM). The optimisation goals include software quality attributes such as performance, reliability and are partitioned into two categories: G+ (goals to be maximized) and G- (goals to be minimized). The ADM represents a Multi Objective Architecture Decision Model (MOADM), which could have a large number of optimization goals. To simplify this model, the MOADM is converted into a cost-benefit decision model (CBDM), establishing the relation among design choices and levels of goal satisfaction. The goal weights and preferences functions are elicited using information provided by stakeholders using MCDA techniques. The approach does not cover dynamic update of utility preferences. The goal preferences are defined as linear functions, where preference 0 and 1 are associated to the lowest and highest possible values for that goal among all candidate architectures and all possible parameters values. The approach takes into account the risk associated with each candidate (design) decision. Adaptation implies shortlisting the candidate design decisions. The default is to shortlist candidate design decisions that maximize expected net benefit and minimize the risk of project failure. An extension of Pareto optimal technique is used for shortlisting. The approach does not support autonomic reasoning as the set of final alternatives must be presented to stakeholders for the final decision.

Garcia-Galan et.al. (AP3) In [17], Garcia-Galan et.al. propose a preference-based analysis for identifying service configurations that maximize the tenants satisfaction at runtime. The approach is inspired in game theory and social adaptation concepts (i.e., it considers changes in the users collective judgment as a new adaptation driver). The analysis made to identify the best system configuration(s) interprets the problem as a coalitional game where all the tenants are considered a great coalition. Conceptually, the problem is solved by the Nash Bargaining Solution [18] and is implemented using multi-objective optimization: a Pareto efficient solution maximizing the Nash Product, this allows shortlisting alternatives. Because there may be many Pareto optimal solutions, in order to choose a single solution it is applied a weighted Nash Product, where the weight sign the importance of each tenant: the more users a tenant presents, the more importance the tenant will have. The approach involves the creation of an

Extended Feature Model (EFM) and a Semantic Ontology of User Preferences (SOUP) [17] preference Model. The EFM is a variability model and represents the configuration space (functional features and quality attributes). SOUP contains the tenants' preferences, called preference terms, which refers to functional and NFRs). The users preferences can be elicited and updated by asking for explicit users feedback or mining the quality feedback from the users behavior. Crucially, this last characteristic could allow an autonomic update of preferences.

Elahi et.al. (AP4) Elahi et.al. [19] present a trade-off analysis algorithm that takes pair-wise comparison of alternative solutions to determine the best solution among several alternatives. It is important to highlight that the comparisons are not transformed into a numerical representations of utility preferences. Valid satisfaction levels that the requirements may have are enumerated with respect to the relative rankings of alternatives. Stakeholders' preferences are incorporated using a MCDA-based method: the Even Swaps Method [20], therefore avoiding the elicitation of numerical weights or goals. There is also support for preference update by asking for explicit users' feedback: i.e. the preference updating is not autonomous. Qualitative parameters (i.e. no numerical) act as input for the reasoning engine. To determine the best alternative for each possible goal satisfaction level, the algorithm decides the optimum alternative by using a heuristic method. When the founded alternative does not satisfy all the NFR an optimum alternative is not found and it is an exceptional pattern. In such a case the domain experts may be consulted.

Liaskos et.al. (AP5) In [1], Liaskos et al. present a goal modelling extension to support preferences. The goals are classified as either preference goals or mandatory. The preference goals are used for evaluating alternative ways to achieve mandatory goals. The approach involves a framework for specifying preferences requirements and priorities among them. These preferences are used by a preference-based planner to search for alternatives which satisfy mandatory and preferred requirements. The weights are assigned to preference goals using a quantitative requirement prioritization scheme: Analytic Hierarchical Process (AHP) [14]. The adaptation is performed formalizing a visual goal model and the preferences specification into a planning problem specification. To have a clear semantics for preferences and priorities the authors use a formal language for specifying AI planning problems called PDDL 3.0 allowing the use of semantics for preferences and their priorities. The reasoning engine was implemented using Hierarchical Task Networks (HTNs): an automated planning, where dependencies among different design alternatives are shown. The planner is implemented using HTNPlan-P [21] and the input for this planner is an HTN domain specification and a set of PDDL preferences and metrics. There are no specific references regarded to preferences updating in runtime.

Peng et.al. (AP6) Peng et.al. [22] propose a self-tuning method that can dynamically tune the preferences of different quality requirements to autonomously make trade-off of decisions by a preference-based goal reasoning procedure. Goals can be either hard or soft goals. The system quality attributes correspond to soft goals. Each soft goal has a preference rank. The preference rank is defined by

a priority number indicating its importance with respect to other soft goals. The decision process involves three main components: (i) a Proportional Integral Derivative (PID) Controller [23], (ii) a Preference-driven Goal Reasoner (GR) and (iii) an Architecture Configurator (AC). In this approach, the quality attribute preferences are updated autonomously by the PID controller, establishing a balance in runtime, between the earned business value and quality measurement. The PID controller dynamically adjust the preference ranks of related soft goals, to maximize the overall satisfaction base on runtime feedback (earned business value and quality measurements) and using a preference tuning algorithm. The algorithm returns the tuned preferences ranks (given between 1 and 10) of soft goals. The reasoning approach uses a Pareto Optimal algorithm that yields a set of alternative configurations and the SAT solver obtain the optimal solution. The GR takes as inputs the soft goals preference ranks and the goal model, with hard and soft goals, the refinement and contribution relationships, and its quantitatives expected satisfaction. The GR first encodes the goal model elements into Conjunctive Normal Form proposition formulas to feed a Satisfiability (SAT) solver. Then, the SAT solver is invoked to get a valid configuration. The GR always finds the Pareto Optimal solutions.

Bencomo et.al. (AP7) In [5], Bencomo et.al. propose the Bayesian definition of surprise as the basis for quantitative analysis to measure deviations of self-adaptive systems from the specified expected behavior. The authors use surprises to measure how observed data affects the assumptions of the world during the execution of the SAS at runtime. In [5], a surprising event is defined as one that causes a large divergence between the belief distributions prior to and posterior to the occurrence of the event. In other words, a surprise can quantify the unanticipation of new data observation given the current model of the environment. When a surprise is perceived, the SAS may decide either to adapt accordingly or to flag that an abnormal situation is happening. AP7 is based on the mathematical model provided by Dynamic Decision Networks (DDNs) [24] which is driven by levels of satisficement of NFRs (i.e. quality priorities). Decisions in the DDN correspond with alternative design decisions dj that correspond with different configurations a SAS can undertake. The random variables of the DDN represent the levels of satisficement of quality properties given the corresponding configurations. The conditional probability $P(NFRi|dj)$ represents the probability of NFRi being satisficed given a decision dj . For each NFRi, a function of the utility preference called w(NFRi ,dj) over every design alternative dj considered is identified. The function w(NFRi,dj) represent the utility preferences. Crucially, the conditional probabilities will be updated by the DDN by Bayesian updating. Probabilistic inference occurs whenever new evidence arrives. The initial conditional probabilities are either estimated by experts or collected from previously gathered statistics. It is important to highlight that the DDN provides a quantitative technique to make informed decisions based on the arrival of new evidence during runtime. The approach allows to uncover conflicts between quality properties and support reasoning about these conflicts and therefore the re-appraisal of their tradeoff due to runtime evidence. Utility

preferences could be updated at runtime based on the new information gathered as shown in our own approach.

Sousa et.al. (AP8) In [25], Sousa et.al. propose a model for capturing user preferences with respect to quality of service (QoS) at design time. User preferences are expressed as independent utility functions for each aspect or dimension of QoS and the model was applied to a case study about resource adaptation for improving user satisfaction with respect to running software on small mobile devices. Using the proposed approach, it is possible to model quality of service tradeoffs based on utility theory. The utility functions map the possible quality levels in a dimension of QoS to a normalized utility space U [0,1], where the user is happy with utility values close to 1, and unhappy with utility values close to zero. The approach define a software infrastructure that captures models of QoS trade-offs, coordinates the resource usage across the applications and enables the applications to dynamically adjust their adaptation policies based on the QoS models. A Solver determines the tactic with the highest utility, given the available resources, by exhaustive evaluation of all the tactics defined for the application. The Solver is invoked by the application before carrying out each unit of work.

Ramirez et.al. (AP9) Ramirez et.al. [26] present a goal-based requirements model-driven approach for automatically deriving utility functions for RELAXed goal models during the requirement engineering phase. The proposed approach was applied to a case study based on the goal model of an intelligent vehicle system (IVS) and uses the derived utility functions to monitor the IVS under different environmental conditions at run time. The approach, called Athena, uses as input a goal model and a mapping between environmental conditions and the monitoring elements responsible for observing them. The approach produces as output a set of different kind of utility functions by instantiating utility functions templates based on the goal's type. For a RELAXed goal, Athena generates a fuzzy logic-based utility function by mapping RELAX operators to corresponding fuzzy logic-based mathematical functions.

Walsh et.al. (AP10) In [27] Walsh et.al. demonstrate how utility functions can enable a collection of autonomic elements to continually optimize the use of computational resources in a dynamic, heterogeneous environment. This is achieved by providing a uniform means of communicating resource needs to a resource arbiter. The approach uses a user interface for modifying utility functions at run-time, therefore there is no autonomic update of preferences at runtime.

Angelopoulos et.al. (SEAMS 2014) (AP11) Angelopoulos et.al. [28] use concepts of goal-oriented requirements engineering, such as goals for modelling stakeholder requirements, soft goals for modelling NFR, Awareness Requirements (AwReqs) and Evolutionary Requirements (EvoReqs). Analytic Hierarchy Process (AHP) is used at design time to support the prioritization of soft goals and AwReqs. The decision making framework used, Zanshin, support a feedback controller that monitors failures and determines the adaptation actions to be carried out. The framework perform adaptations by using reconfigurations. Reconfiguring consists on finding an alternative specification with new values for

system parameters in order to improve indicators of failure (i.e. AwReqs) and take the system back to an acceptable state. The specific mechanism for reconfiguration is called Qualia+, which is an adaptation mechanism that defines a basic algorithm composed of 8 procedures such as parameter selection and parameter change. The basic algorithm uses information about AwReqs, parameters and their interrelations in order to randomly select which parameter to tune from the set of parameters that can positively affect a given indicator of a failing in a SAS. The first results were positive in comparison with a previous version of the mechanism of reconfiguration. Next steps will involve further evaluation to ensure the efectiveness on several case studies. The approach does not provide dynamic autonomic update of preferences.

4 Approaches analysis

The results of the analysis and comparison between approaches are shown in Table 3. When the collected information for a possible answer does not fit with any of the research questions, this has been left blank represented by a dash (-). Because the reasoning process in SASs and the system goals are closely related, it is relevant to classify the goal representation and reasoning techniques of the approaches. Therefore, it has been defined a classification criteria about goals and decision making: (CC1) Decision making techniques. (CC2) Goals and quality attributes representation. (CC3) Case study/application. As the specification techniques for quality attribute preferences are the focus of this review, further classification criteria are set with respect to preferences: (CC4) Techniques for design time (CC5) Support for preference updating. (CC6) Techniques for preference updating. These criteria allow the identification of relevant techniques for specifying and reassessing preferences in SAS to further determine those approaches used at design time and runtime. The analysis is explained by using categories which include the classification criteria previously defined.

4.1 Terminology (CC2)

There is not common terminology to refer quality attributes of a system. Song et. al. [15] call them weak constraints and Liaskos et. al. preference goals [1]. In [22], the quality attributes correspond to soft goals and in [17] they are called preferences, which refers to functional and NFR. However the most widespread common denominator is to recognize them as requirements [5, 27, 17], specifically NFR to refer quality attributes.

4.2 Techniques for Design Time (CC4)

The most widespread method for preference elicitation is done by asking the stakeholders of a system. In [15, 5] preferences are elicited initially with information from system stakeholders. Authors in [16, 19, 1] complement this approach by using multi criteria decision analysis methods (MCDA). Elahi et. al. [19] are incorporating Stakeholders preferences by using incorporated using a MCDA-based method: the Even Swaps Method [20], therefore avoiding the elicitation of numerical weights. The most common MCDA methods is AHP [14]. Garcia

Galan et. al. [17] defined the Systems initial preference by using initial configurations defined by the stakeholders. While it is not common, some authors have defined preferences in an autonomous way, for example, Ramirez et. al. [26] use predefined models to derive utility functions.

4.3 Techniques for Preference Updating at runtime (CC5, CC6)

This is a field that need more exploration, several approaches [1, 16, 26] only work with the initial preference at runtime: they do not support preference updating, and some other support it but not in an autonomous way as they required a user intervention: In [15], if users do not agree with the final solution, they can revise the configuration values. The new users' preferences elicited allow tuning the weights of existing goals or the generation of new ones. [25] use a user interface for manipulating thresholds on preferences at runtime. However, Peng et. al. [22] show first results of autonomous preference updating by monitoring the environment at runtime and using a preference tuning algorithm. Bencomo et. al [5] identifiy at runtime the need of preference reassessment. In [27], Walsh et. al. use machine learning techniques to update the system's preferences.

4.4 Decision making techniques (CC1)

Preferences are used by the system in its decision making process. The most common approaches for decision making are utility functions and Pareto Optimal. Ramirez et. al. [26] performs a goal model transformation into utility functions for monitoring software requirements at runtime. In [22], the quality attribute preferences are updated autonomously establishing, at runtime, a balance between the earned business value and quality measurement. The reasoning approach uses a Pareto Optimal algorithm that yields a set of alternative configurations and the satisfiability solver (SAT) obtain the optimal solution. Some approaches have scalability issues, that is, it is not yet possible to easily apply them to real domain problems. Authors in [15] are still investigating how to deal with scalabilities issues associated with the number of constraints. In [17], the approach suffers from scalability problems with respect to the size of its configuration space. Approaches like Sousa et. al. [25], use forecast and learning elements for the decision making process

4.5 Application Domain (CC3)

The approaches have been applied to at least one software application related to an application domain. In [1], the authors tested four applications, therefore this is the approach with the most diverse application domains tested. A summary of this criterion for all the approaches is shown in in Table 3.

5 Findings and concluding remarks

The results show that, even if scarce, there have been important research results towards decision-making for SAS taking into account quality attributes (this is a good aspect). However, dynamic update of utility preferences is still a challenge. This study also shows that preferences specification is still a task executed primarily at design time. The latter are bad aspects and, they are bad and not

Table 3. Approaches summary

	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	AP9	AP10	AP11	Our approach
Criteria												
CC1	CSP/shortest path algorithm	Cost/Benefit - Pareto Optimal	Game Theory/Pareto Nash Product	Heuristic Method	HTNPlan-P planner	CNF/Pareto Optimal	DDN/Mathematical DDN model	A solver for decision making	Goal model transformation into utility functions	Algorithms based on mixed-integer programming	Parameterized adaptation	DDN/Mathematical DDN model
CC2	Weak constraints	Software quality attributes	Nonfunctional terms	NFR	Preference goals	Soft Goals	NFRs	User preferences	Functional invariant goals and non-invariant goals	Service level attributes and Resource Level Attributes	Soft-goals	NFRs
CC3	✓ (1 application)	✓ (1 application)	✓ (1 application)	✓ (1 application)	✓✓ (several applications)	✓ (1 application)	✓✓ (2 different applications)	✓ (1 application)	✓ (1 application)	✓ (1 application)	✓ (1 application)	✓✓ (2 different applications)
CC4	Stakeholders' feedback	MCDA Techniques	Stakeholders' feedback	MCDA method: Even Swaps	MCDA method: AHP	Preference tuning algorithm	Experts/statistics	Collected from users	Derived from a RELAXed Goal Model	Service level utility function based on business value expressed by stakeholders	AHP applied to soft-goal models	MCDA method: P-CNP
CC5	✓	-	✓	✓	-	✓	+	✓	-	✓	-	+
CC6	Updating with user intervention	-	Users' feedback/mining users' behavior	By asking for users' feedback	-	Preference tuning algorithm	Bayesian surprise	Using a user interface	-	By asking for users' feedback	-	Bayesian surprise & NFRs satisfaction Level

Key

- When the collected information is not enough to answer a question or it is not offered

CC3 ✓ the approach has been applied in one application ✓✓ the approach have been applied in several different applications

CC5 ✓ the approach supports dynamic preference update + the approach is in progress for supporting dynamic preference update

an ugly as we have found that there are emerging results that proof that challenges are continuously met. To name some emerging results we have shown hat different MCDA techniques stand out as common techniques used for reasoning optimization. It is Pareto, in different variants, one of the most widely spread used [16, 17, 22]. Some approaches use ad-hoc methods for collecting users' preferences, while others use techniques such as MCDA [16, 19, 1]. In [15, 19, 27] the support for preferences update exists but requires user intervention. Some approaches offer potential to support autonomic preference updating. For example, authors of [17] propose an approach for mining users' behavior while authors of [1] use an autonomic preference tuning algorithm. In [26], there is an autonomic generation of utility functions. Authors in [27, 29] highlight the relevance of using models that need to be learned and refined during the operation of the system. Thus, coming back to our initial aim of identifying the existing studies that explicitly consider the need to reassess utility preferences at runtime, the current results still sign there is an important research gap. We are currently working towards this research issue[30]. As shown by experiments discussed in [5] and our recent approach for quality attribute preferences reassessment presented in [30, 4], utility preferences associated with quality attributes initially provided by domain experts during the sensitivity analysis process at design time, may not be ideal for given specific cases found at runtime. Badly chosen preferences can either make the system miss adaptations or suggest unnecessary adaptations that may degrade the behaviour of the running system. To our knowledge, currently there is no significant related work with respect to this specific research issue. Specifically, we are working towards to answering the questions: how to detect quality attribute utilities that are not suitable to the new contexts and which can either trigger non-suitable adaptations or miss adaptation opportunities given changes related to dynamic contexts during execution and that were not fully foreseen before runtime? how to optimize and get scalable reasoning techniques including dynamic updating for quality attribute preferences? We are working on a novel approach, called ARRoW, for the re-appraisal and further updating of the weights associated with quality attributes during runtime and, according to evidence gathered from the operational environmental. The result is a better informed decision-making process based on evidence. Some analyzed approaches [1], including our current research [5, 31], use Artificial Intelligence and machine learning techniques to support the MAPE-K control loop in SAS. Machine learning techniques offer a promising path to be able to explore the operating environment to improve its understanding and get a better informed decision-making [31]. One proposed way is to identify new mechanisms that quantify the difference between the SAS expected behavior and possible deviations due to changes in the environment. It is argued that this quantification must be independent of the current quality attribute utilities assigned by experts. The detected deviations would be triggered in an independent way from the preferences of stakeholders. An example is shown in [5], where it is proposed the use of Bayesian surprises to support a review process of sensitivity analysis to agree on consistent utility functions to new. Furthermore, the idea of surprise

does not have to be attached to Bayesian reasoning only [32]. These new envisioned techniques have the potential of being used, in general, as ways to review and re-asses the sensitivity analysis of the utilities initially assigned. An important consequence is the ability to uncover conflicts between different quality attributes and support reasoning about these conflicts and therefore, allow the re-appraisal of their tradeoff due to evidence found during runtime (supported for example by machine learning) [5]. To achieve the above, it is argued that more collaborations with different communities (AI, MCDA and Search Based Software Engineering techniques, Genetic Programming, etc.) should be performed [8, 33, 34]. The latter is part of the "ugly" aspect. A big question is how do we put in contact all these communities to work together towards a common goal? This is a big challenge!

References

1. S. Liaskos, S. A. McIlraith, S. Sohrabi, and J. Mylopoulos, "Representing and reasoning about preferences in requirements engineering," *Requir. Eng.*, 2011.
2. C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive and Mobile Computing*, vol. 17, Part B, pp. 184 – 206, 2015.
3. N. Bencomo and A. Belaggoun, "Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks," in *REFSQ* -, 2013.
4. L. H. G. Paucar and N. Bencomo, "Runtime Models Based on Dynamic Decision Networks : Enhancing the Decision-making in the Domain of Ambient Assisted Living Applications," *MRT - Models@run.time at MODELS 2016*, 2016.
5. N. Bencomo and A. Belaggoun, "A world full of surprises: bayesian theory of surprise to quantify degrees of uncertainty," in *ICSE*, 2014, pp. 460–463.
6. M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Trans. Auton. Adapt. Syst.*, vol. 4, no. 2, May 2009.
7. B. H. Cheng and et al., "Software engineering for self-adaptive systems." Springer-Verlag, 2009, ch. Software Engineering for Self-Adaptive Systems: A Research Roadmap.
8. R. de Lemos, H. Giese, H. Müller, and M. Shaw, "Software Engineering for Self-Adpaptive Systems: A second Research Roadmap," in *Software Engineering for Self-Adaptive Systems*, ser. Dagstuhl Seminar Proceedings, Germany, 2011.
9. M. Salama, R. Bahsoon, and N. Bencomo, "Managing trade-offs in self-adaptive software architectures: A systematic mapping study," in *Managing trade-offs in adaptable software architectures*, I. Mistrk, N. Ali, J. Grundy, R. Kazman, and B. Schmerl, Eds. Elsevier, 2016.
10. B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds., *Software Engineering for Self-Adaptive Systems*, ser. LNCS, vol. 5525. Springer, 2009.
11. A. Ishizaka and P. Nemery, "Multi-criteria decision analysis : Methods and software." *Somerset, NJ, USA: John Wiley & Sons*, 2013.
12. J. Figueira, S. Greco, and M. Ehrogott, *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer, 2005.
13. M. P. D. S. J. . Y. S. Harman, M., "Search based software engineering: Techniques, taxonomy, tutorial." *Search, 2012, 159.*, 2011.
14. T. Saaty, "Decision making with the analytic hierarchy process," *Inter. Journal of Services Sciences,*, 2008.

15. H. Song, S. Barrett, A. Clarke, and S. Clarke, "Self-adaptation with end-user preference: Using run-time models and constraint solving," in *the Intrl. Conference MODELS*, USA, 09/2013 2013.
16. E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *Proceedings of ICSE*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 883–894.
17. J. García-Galán, L. Pasquale, P. Trinidad, and A. Ruiz-Cortés, "User-centric adaptation of multi-tenant services: Preference-based analysis for service reconfiguration," in *SEAMS*, ser. SEAMS 2014. USA: ACM, 2014, pp. 65–74.
18. J. F. Nash, "The bargaining problem," *Econometrica: Journal of the Econometric Society*, 1950.
19. G. Elahi and E. Yu, "Requirements trade-offs analysis in the absence of quantitative measures: A heuristic method," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11. New York, NY, USA: ACM, 2011.
20. K. R. L. R. H. Hammond, John S., "Even swaps: A rational method for making trade-offs," *Harvard Business Review, MarchApril, 137150*, 1998.
21. B. J. a. . M. S. a. Sohrabi, S., "Htn planning with preferences," *IJCAI International Joint Conference on Artificial Intelligence, 17901797.*, 2009.
22. X. Peng, B. Chen, Y. Yu, and W. Zhao, "Self-tuning of software systems through goal-based feedback loop control," in *Requirements Engineering Conference (RE)*, Sept 2010, pp. 104–107.
23. J. D. P. G. F. Franklin and A. E. Naeini, "Feedback control of. dynamic systems, 5th ed." *Upper Saddle River, NJ, USA: Prentice-Hall*, 2006.
24. N. Bencomo, A. Belaggoun, and V. Issarny, "Dynamic decision networks to support decision-making for self-adaptive systems," in *(SEAMS)*, 2013.
25. J. P. Sousa, R. K. Balan, V. Poladian, D. Garlan, and M. Satyanarayanan, "User guidance of resource-adaptive systems," in *In Proc. of International Conference on Software and Data Technologies*, 2008.
26. A. J. Ramirez and B. H. C. Cheng, "Automatic derivation of utility functions for monitoring software requirements," *Lecture Notes in Computer Science*, vol. 6981 LNCS, pp. 501–516, 2011.
27. W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das, "Utility functions in autonomic systems," in *Autonomic Computing*, 2004.
28. K. Angelopoulos, V. E. S. Souza, and J. Mylopoulos, "Dealing with multiple failures in zanshin: A control-theoretic approach," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS 2014. New York, NY, USA: ACM, 2014, pp. 165–174.
29. N. Bencomo, A. Bennaceur, P. Grace, G. Blair, and V. Issarny, "The role of models@run.time in supporting on-the-fly interoperability," *Computing*, vol. 95, no. 3, pp. 167–190, 2012.
30. L. H. G. Paucar and N. Bencomo, "The Reassessment of Preferences of Non-Functional Requirements for Better Informed Decision-making in Self-Adaptation," *AIRE '16 at RE' 2016.*, 2016.
31. S. Hassan, N. Bencomo, and R. Bahsoon, "Minimize nasty surprises with better informed decision-making in self-adaptive systems," in *SEAMS*, 2015.
32. N. Bencomo, "Quantun: Quantification of uncertainty for the reassessment of requirements," in *RE*, 2015, pp. 236–240.
33. M. Harman, P. McMinn, J. T. de Souza, and S. Yoo, "Empirical software engineering and verification," B. Meyer and M. Nordio, Eds. Springer-Verlag, 2012.
34. C. M. B. B. H. Cheng, "An approach to mitigating unwanted interactions between search operators in multi-objective optimization," *GECCO '15*, 2015.