A Protocol for a Classification Framework of Uncertainty in Architecture-based Self-Adaptive Systems with Multiple Quality Requirements

Sara Mahdavi-Hezavehi

Paris Avgeriou

Danny Weyns

Software Architecture and Engineering group – University of Groningen

AdaptWise research group – Linnaeus University

Table of Contents

Introduction

Software systems operating in changing environments need human supervision to adapt to changes. The need for adaptation in a software system may stem from a variety of anticipated and unforeseen factors including, but not limited to, new stakeholders' requirements, changes in requirements priorities, and changes in the environments. As the complexity of these software systems increases, so does their supervision overhead. Therefore, one of the main goals during design and implementation of such systems is to decrease the amount of human involvement in a running self-adaptive system is one of the main goals of designing and implementing of these systems. The idea behind an adaptive software system is to design and develop a system capable of autonomously adapting to changing conditions at runtime. This autonomous adaptability decreases the cost and operation time required for adapting the system while fulfilling certain quality requirements at runtime. Although high complexity and heterogeneity of software systems inhibit design and development of such software systems, many promising self-adaptation methods[1] have been developed and evaluated in academic settings in the past decade. However, uncertainty in the self-adaptive software system has been obstructing the application of these methods in real-world software systems (Esfahani & Malek, 2013).

Uncertainty may be due to dynamicity of the environment, changing systems goals, lack of adequate knowledge at the design time, or an aftermath of applying adaptation actions to the system at runtime; defective adaptation actions or unforeseen consequences of adaptation on system requirements can result in unexpected system behavior. This is further aggravated in self-adaptive systems with multiple quality requirements, which need to simultaneously fulfil multiple quality requirements without interrupting the system's normal functions. This implies that the system should be able to prioritize the adaptation actions, choose the optimal adaptation scenarios, adapt the system, and presumably handle the positive or negative chain of effects caused by the adaptation of certain requirements. This assures that adaptation does not impede the system's overall quality attributes and functionalities. However, when the number of system concerns increases, so does the number of representing adaptation alternatives. Therefore, the decision making, as well as the handling of requirements trade-offs becomes more complex.

1. Background

In the following section, we present a brief description for self-adaptive systems, self-adaptive systems with multiple quality requirements, uncertainty in self-adaptive systems, and architecture-base self-adaptation.

Self-adaptive systems: Self-adaptive systems are capable of modifying their runtime behavior in order to achieve systems objectives. Unpredictable circumstances such as changes in the system's environment, system faults, new requirements, and changes in the priority of requirements are some of the reasons for triggering adaptation action in a self-adaptive system. To deal with these uncertainties, a self-adaptive system continuously monitors itself, gathers data, and analyzes them to decide if adaption is required. The challenging aspect of designing and implementing a self-adaptive system is that not only must the system apply changes at runtime, but also fulfill the systems requirements up to a satisfying level. Engineering such systems is often difficult as the available knowledge at design time is not adequate to anticipate all the runtime conditions. Therefore, designers often prefer to deal with this uncertainty at runtime, when more knowledge is available.

---

1 In this document "self-adaptation method" refers to any type of solution(s) proposed to handle requirements in domain of self-adaptive systems.

Architecture-based self-adaptation: Architecture-based self-adaptation (Oreizy et al., 1998) is one well recognized approach that deals with uncertainties by supporting modifiable runtime system behavior (Garlan et al., 2004). Architecture-based self-adaptation realizes adaptation through the MAPE-K (i.e., Monitor, Analyze, Plan, Execute, and Knowledge component) reference model (IBM, 2005). By complying with concept of separation of concerns (i.e. separation of domain specific concerns from adaptation concerns), the MAPE-K model supports reusability and manages the complexity of constructing self-adaptive systems. This makes the MAPE-K model a suitable reference for designing feedback loops and developing self-adaptive systems. Nonetheless, the majority of current approaches dealing with uncertainty not only makes use of MAPE-K reference model, but also takes advantage of probability and possibility theories. One well-known architecture-based self-adaptive framework is Rainbow (Garlan et al., 2004). Rainbow uses an abstract architectural model to monitor software system runtime specifications, evaluates the model for constraint violations, and if required, performs global or module-level adaptations. (Calinescu et al., 2011) present a quality of service management framework for self-adaptive services-based systems which augments the system architecture with the MAPE-K loop functionalities. In their framework, the high-level quality of service requirements are translated into probabilistic temporal logic formulae which are used to identify and enforce the optimal system configuration while taking into account the quality dependencies. Moreover, utility theory can be used (Cheng et al., 2006) to dynamically compute trade-offs (i.e. priority of quality attributes over one another) between conflicting interests, in order to select the best adaptation strategy that balances multiple quality requirements in the self-adaptive system.

Architecture-based self-adaptive systems with multiple quality requirements: Similar to any other software system, architecture-based self-adaptive systems should fulfill a variety of quality attributes in order to support a desired runtime system behavior and user experience. To design and develop such self-adapting systems, it is important to analyze the tradeoffs between multiple quality attributes at runtime, and ensure certain quality level after adaptation actions. This means that not only requirements with higher priorities, which define the system's goal, should be met; but also quality attributes of the system should be fulfilled at an acceptable level. After all, a systems' overall quality is a desired combination of several runtime and design time requirements. However, majority of current architecture-based self-adaptive systems approaches fail to address the negative impacts of adaptation method on multiple quality attributes, and this issue deteriorates systems' overall quality in complex software systems.

Uncertainty in architecture-based self-adaptive systems: Uncertainty in an architecture-based self-adaptive system or self-adaptive systems in general, can be studied from a number of different perspectives. First and foremost genre of uncertainty is caused by dynamicity and unpredictability of a variety of factors existing in software systems. In fact, this type of uncertainty justifies the need for design and development of self-adaptive systems. An architecture-based self-adaptive system should be able to investigate solutions space, choose the optimal adaptation action, and adapt the system while fulfills quality requirement of the system in a satisfying level. However, in a system with multiple objectives, and quality goals the decision making process for selecting the optimal adaptation action is quite complex; which leads us to the second genre of uncertainty in architecture-based self-adaptive systems: consequences of self-adaptation in a software system. Incorporating self-adaptation capability into a software system may produce even more complexity and undesirable effects in the system. Not only the self-adaptive system should deal with a growing solution space for adaptation, but it also needs to handle possible negative effects of adaptation on the system. Adversely affecting quality requirements of the system, noise in sensing and imperfect application of adaptation actions are examples of uncertainties which are aftermaths of self-adaptation in a system. Lastly, the concept of

uncertainty itself and its characteristics are vaguely described and interchangeably used to refer to a variety of notions in domain of architecture-based self-adaptive systems with multiple quality requirements; this poses more ambiguity to the topic of uncertainty in this domain.

1.1. Quality attributes in self-adaptive systems

We provide a list of QAs with their definitions as a guideline for extracting correct and relevant data from the primary studies.

In (Salehie & Tahvildari, 2009) Salehie et al. demonstrate how a set of well-known QAs from ISO 9126-1 quality model can be linked to main self-* properties. They argue that self-configuring potentially impacts several quality factors, such as maintainability, functionality, portability, and usability. According to their study, self-optimizing has a strong relationship with efficiency, and self-protecting can be associated with reliability of the system. Finally, in self-healing, the main goal is to maximize the availability, survivability, maintainability, and reliability of the system (*ISO/IEC 9126-1 Software Eng. -Product quality - Part 1: Qualitymodel, Int. Standard Organization*, 2001).

In (Weyns & Ahmad, 2013) Weyns et al. concluded that efficiency/performance, reliability, and flexibility are the most addressed QAs in domain of self-adaptive systems, and these QAs are claimed to be improved by current self-adaptation methods. In addition, their results indicate that security, accuracy, usability, and maintainability are relevant to self-adaptive systems as well.

Based on aforementioned data, we aggregate the following list of QAs; the definitions are based on ISO/IEC 25012and IEEE9126 ("ISO9126 - Software Quality Characteristics," n.d.), ("ISO/IEC 25012:2008 - Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Data quality model," n.d.):

- **Performance:** efficiency of the software by using the appropriate amounts and types of resources under stated conditions and in a specific context of use.

- **Reliability:** refers to the ability of software to maintain a specified level of performance while running in specified conditions.

- **Flexibility:** capability of the software to provide quality in use in the widest range of contexts of use , including dealing with unanticipated change and uncertainty)

- **Maintainability:** refers to the capability of software to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in the environment, and in functional and non-functional specifications.

- **Availability:** is the ability of software to be in a state to perform required functions at a given point in time, under specified conditions of use. Therefore, it can be considered as a combination of maturity (i.e., ability to avoid failure), fault tolerance and recoverability (i.e., ability to re-establish a specified level of performance after failure)("ISO9126 - Software Quality Characteristics," n.d.).

Note that we do not limit the included primary studies only to those studies addressing the above mentioned QAs; if certain studies address other QAs than what we have listed above, we keep the studies and use them in the data analysis phase as they potentially contain relevant data to our domain of interest.

1.2. Problem statement and Justification

The underlying uncertainty in a self-adaptive system, aggravates the complexity of dealing with a growing number of adaptation scenarios, and requirements trade-offs (Cheng et al., 2006), making the self-adaptive system even more inconsistent, and complicated to function autonomously; and introduces a new level of uncertainty to the system. If not handled properly, overtime, uncertainty provokes inconsistency in certain subsystems, and the accumulated inconsistencies may result in circumstances the self-adaptive system has not been designed to deal with, and as a consequence, the system displays unexpected behavior. Despite the fact that various approaches are available to quantify and mitigate existing uncertainty, the concept of uncertainty, its different types and categories, are vaguely defined and ambiguously understood in the domain of self-adaptive systems (Esfahani & Malek, 2013). As a result, identification, investigation, and consequently selection of suitable approaches for tackling uncertainty are problematic. Therefore, we carry out this systematic study in order to review current architecture-based approaches tackling uncertainty in self-adaptive systems with multiple quality requirements. Furthermore, we aim to identify common sources of uncertainty in systems with multiple quality requirements, and to investigate specifications of current solutions addressing these uncertainties. Finally, we propose a structured classification framework for uncertainty and its characteristics in domain of self-adaptive systems.

2. Research Methodology: Systematic Literature Review

In this study we aim at identifying, exploring, and classifying state of the art on architecture-based methods handling uncertainty in self-adaptive systems with multiple quality requirements. Therefore, we perform a systematic literature review (Kitchenham & Charters, 2007) to collect and investigate existing architecture-based methods, and to answer a set of pre-defined research questions. The first step of conducting a systematic literature review is to create a protocol, in which all the steps and details of the study are specified. In the protocol, we present our research questions, a generic overview of the process and the search strategy which we use to search through selected databases, inclusion and exclusion criteria for filtering the collected papers, data extraction procedure, and methodology we use to synthesize the data to answer the research questions and propose the classification framework.

2.1. Research Questions

We pose the following research questions to investigate the current architecture-based approaches tackling uncertainty in self-adaptive systems with multiple quality requirements.

1. What are the current architecture-based approaches tackling uncertainty in self-adaptive systems with multiple requirements?
2. What are the different uncertainty dimensions which are explored by these approaches?
    a. What are the options for these uncertainty dimensions?
3. What sources of uncertainties are addressed by these approaches?
4. How are the current approaches classified according to the proposed uncertainty classification framework?

By answering research question one, we get an overview of current architecture-based approaches tackling uncertainty. “Architecture-based” implies that the approach presented in the study should provide architectural solutions (e.g., architectural models) to handle and reason about the dynamic behavior of the system. To be more specific, the subject system should use its self-architectural model elements to monitor and adapt its runtime behavior. In other words, it should be possible to map components of systems ‘architectural model (i.e., self-architectural model) to MAPE-k functionalities. In Architecture-based self-adaptive systems with multiple quality requirements, system needs to simultaneously fulfil multiple quality requirements without interrupting the system’s normal functions. Therefore, an architecture-based self-adaptive system is expected to efficiently manage the adaptation actions (i.e., monitor the system, identify the problems, propose solutions, and apply adaptations to the system), tackle potential uncertainties in the system, and meet functional and quality requirements at a satisfying level. By answering this research question, we will obtain e better understanding of existing architecture-based approaches tackling uncertainty in self-adaptive systems with multiple quality requirements. Answer to this research question will be a list of current studies, related venues and books in which they have been published, year of publication, and authors’ names.

Research question two tries to identify and investigate possible dimensions for uncertainty. Dimensions refer to different aspects of uncertainty in self-adaptive systems with multiple quality requirements. For instance, we are interested in figuring out whether or not locations (e.g., environment, adaptation functions) in which the uncertainty manifests itself are a commonly discussed subject, or if phases of systems life cycle in which the existence of uncertainty is acknowledged, etc. are discussed in the selected papers or not. Answer to this research question will help us to derive the most significant and common aspects of uncertainty in this domain.

Research question 2.a tries to analyze the dimensions of uncertainty resulting from the previous research question, on a more concrete level. By answering this research question, we come up with a

list of common categories and options for each of the aforementioned dimensions. For instance, we intend to come up with a list of possible locations in which the uncertainty appears in a self-adaptive system, or to identify in which particular phases of systems life cycle the existence of uncertainty is acknowledged or the problem is tackled.

The source of uncertainty is one of the most important dimensions of uncertainty, so we investigate it in more depth in research questions three. By answering this research question, we try to identify and list common sources of uncertainty, from which the uncertainty originates. Sources of uncertainty refer to a variety of circumstances which affect and deviate system behavior from expected behavior in the future. For example, changes in the environment or systems requirements are considered as sources of uncertainty. The list of sources of uncertainty will be a separate part of the final classification framework. Answers to research questions two, three and four help to compose the classification framework, which is the main contribution of this study.

Finally, we pose research question four to indicate how the proposed uncertainty classification framework can be used to study and classify current approaches tackling uncertainty in domain of self-adaptive systems with multiple quality requirements. We essentially investigate the usefulness of the proposed classification framework by analyzing selected primary studies, and mapping them to the framework.

To sum up, by answering the aforementioned research questions, we present an overview of existing architecture-based approaches tackling uncertainty in self-adaptive systems with multiple requirements. In addition, we strive to identify common dimensions, characteristics of those dimensions, and sources which are treated in the literature, and propose a comprehensible classification framework for uncertainty in self-adaptive systems with multiple quality requirements. Finally, we use the proposed framework as the basis for further analysis of extracted data from the selected papers to present a statistical overview of the current research in this domain.

2.2. Search strategy

Similar to determining a “quasi-gold” standard as proposed by Zhang and Babar (Zhang & Ali Babar, 2010), we manually search a small number of venues to cross check the result we get from automatic search. This helps to create valid search strings. To perform the manual search, we select venues based on their significance for publishing research in the context of self-adaptive systems. For the publication time, we limit the manual search to the period of January first of 2000 and 20th of July of 2014.This is because the development of successful self-adaptive software hardly goes back to a decade ago; after the advent of autonomic computing (Calinescu, 2013). Note that even though some major conferences on self-adaptive systems started to emerge after 2005 (e.g., SEAMS), we chose to start the search in the year 2000 to avoid missing any studies published in other venues. Therefore, we manually search the following venues:

-International Conference on Software Engineering
-Software Engineering for Adaptive and Self-Managing Systems
-Transactions on Autonomous and Adaptive Systems

In the manual search, we consider title, keywords, and abstract of each paper. After finishing the manual and automatic search for the aforementioned venues, we can compare the results to get an estimation of the coverage of the automatic search. The results from the automatic search should include all studies found for the “quasi-gold” standard (i.e., the “quasi-gold” standard should be a subset of the results returned by the automatic search).

2.2.1. Search method

We use automatic search to search through selected venues. By automatic search we mean search performed by executing search strings on search engines of electronic data sources. Although manual search is not feasible for databases where the number of published papers can be over several thousand(Ali, Ali Babar, Chen, & Stol, 2010), we are still incorporating a manual search (i.e., "quasi-gold" standard) into the search process to make sure that the search string works properly. We include any type of study (empirical, theoretical, etc.) as long as it is relevant to the research domain.

2.2.2. Search terms for automatic search

One of the main challenges of performing an automatic search to find relevant studies in the domain of self-adaptive systems is a lack of standard, well-defined terminology in this domain. Due to this problem, and to avoid missing any relevant paper in the automatic search, we prefer to use a more generic search string and include a wider number of papers in the primary results. Later, we will filter out the irrelevant studies to get the final papers for data extraction purpose. We used the research questions and a stepwise strategy to obtain the search terms; the strategy is as follows:

1. Derive major terms from the research questions and the topics being researched.
2. If applicable, identify and include alternative spellings, related terms and synonyms for major terms.
3. When database allows, use "advance" or "expert" search option to insert the complete search string.
    a. Otherwise, use Boolean "or" to incorporate alternative spellings and synonyms, and use Boolean "and" to link the major terms.
4. Pilot different combinations of search terms in test executions.
5. Check pilot results with "quasi-gold" standard.
6. Discussions between researchers.

As a result, following terms are used to formulate the search string:

> Self, Dynamic, Autonomic, Manage, Management, Configure, Configuration, Configuring, Adapt, Adaptive, Adaptation, Monitor, Monitoring, Heal, Healing, Architecture, Architectural

The search string consists of three parts, Self AND Adaptation AND Architecture. The alternate terms listed above are used to create the main search string. This is done by connecting these keywords through logical OR as follow:

> (self **OR** dynamic **OR** autonomic) **AND** (manage **OR** management **OR** configure **OR** configuration **OR** configuring **OR** adapt **OR** adaptive **OR** adaptation **OR** monitor **OR** monitoring **OR** analyze **OR** analysis **OR** plan **OR** planning **OR** heal **OR** healing **OR** optimize **OR** optimizing **OR** optimization **OR** protect **OR** protecting) **AND** (architecture **OR** architectural)

The reference search string may go through modifications due to search features of electronic sources (e.g., different field codes, case sensitivity, syntax of search strings, and inclusion and exclusion criteria like language and domain of the study) provided by each of the electronic sources.

2.2.3. Scope of search and sources to be searched

The scope of the search is defined in two dimensions: publication period (time) and venues. In terms of publication period, we limited the search to papers published over the period first of January of 2000 and 20th of July of 2014 as mentioned in 2.2.

For each venue, we document the number of papers that was returned. Also, we record the number of papers left for each venue after primary study selection on the basis of title and abstract. Moreover, the number of papers finally selected from each venue should be recorded. This information are mainly recorded for documentation purposes, but may also be used later in the data analysis phase. The venues to be searched are shown in Table 1.

Table 1 – List of venues to be search automatically.

Conference proceedings and Symposiums	International Conference on Software Engineering (ICSE)
	IEEE Conference on Computer and Information Technology (IEEECIT)
	IEEE Conference on Self-Adaptive and Self-Organizing Systems (SASO)
	European Conference on Software Architecture (ECSA)
	International Conference on Autonomic Computing (ICAC)
	International Conference on Software Maintenance (CSM)
	International Conference on Adaptive and Self-adaptive Systems and Applications (ADAPTIVE)
	Working IEEE/IFIP Conference on Software Architecture (WICSA)
	International Conference of Automated Software Engineering (ASE)
	International Symposium on Architecting Critical Systems (ISARCS)
	International Symposium on Software Testing and Analysis (ISSTA)
	International Symposium on Foundations of Software Engineering (FSE)
	International Symposium on Software Engineering for Adaptive & Self-Managing Systems (SEAMS)
Workshops	Workshop on Self-Healing Systems (WOSS)
	Workshop on Architecting Dependable Systems (WADS)
	Workshop on Design and Evolution of Autonomic Application Software (DEAS)
	Models at runtime (MRT)
Journals/Transactions	ACM Transactions on Autonomous and Adaptive Systems (TAAS)
	IEEE Transactions on Computers (TC)
	Journal of Systems and Software (JSS)
	Transactions on Software Engineering and Methodology (TOSEM)
	Transactions on Software Engineering (TSE)
	Information & Software Technology (INFSOF)
	Software and Systems Modeling (SoSyM)
Book chapters/Lecture notes/Special issues	Software Engineering for Self-Adaptive Systems (SefSAS) Software Engineering for Self-Adaptive Systems II (SefSAS)
	ACM Special Interest Group on Software Engineering (SIGSOFT)
	Assurance for Self-Adaptive Systems (ASAS)

To get the list of venues for automatic search, we have included the list of venues searched by D.Weyns at al. in (Weyns & Ahmad, 2013). In their systematic literature review they included a list of high quality primary studies in domains of self-adaptive systems, software architectures, and software engineering. Furthermore, to broaden the search scope, we used Microsoft Academic Search[2] to find more relevant venues in domain of self-adaptive systems, and software architectures and included them in the study.

[2] . http://academic.research.microsoft.com/

To perform the automatic search of the selected venues, we use a number of most relevant electronic sources to software engineering listed in (Kitchenham & Charters, 2007):

1. IEEE Xplorer
2. ACM digital library
3. SpringerLink
4. ScienceDirect

Not only these libraries are most relevant to the field of study, but also they can cover most of the selected venues. In addition, they provide fairly powerful and easy to use search engines which are suitable for automatic search of data bases.
We are aware of the fact that a few of the venues may not be accessible through any digital library. This is an exception, and we plan to manually search these particular venues in the searching phase.

2.2.4. Search process

We adapted the search process of (Mahdavi-Hezavehi, Galster, & Avgeriou, 2013) and use a four phased searching process to collect the final papers (Figure 1). In the first phase, we manually search the venues listed in 2.2 to establish the "quasi-gold" standard. To perform the manual search we look into papers' titles, keywords, abstracts, introductions, and conclusions. In addition, we record the total number of papers we looked at (per venue), number of relevant papers returned, and number of papers imported to reference manager tool (per venue). This phase results in a set of papers which later on must be a subset of automatic search results. In the next phase, we perform the automatic search of selected venues listed in 3.2.3. Depending on the search engines' options, two different searching strategies can be selected. If the search engine allows searching the whole paper, we use the search string to search the full paper; otherwise, titles, abstracts and keywords must be searched. In this phase, the search string used to perform the search, search settings, number of returned papers, and the number of imported papers to the reference manager tool should be recorded (all factors should be stored per source searched). Next, we enter the filtering phase in which, we filter the results based on titles, abstracts, keywords, introductions, and conclusions, and also remove the duplicate papers. This results in a set of potentially relevant papers, and should be compared to the "quasi-gold" standard. If the condition holds (i.e. "quasi-gold" papers are subset of automatic results), we start filtering the papers based on inclusion and exclusion criteria to get the final set of papers. In this phase, the inclusion/exclusion criteria (per paper), number of remaining papers, and list of remaining papers should be recorded. In this phase, we simultaneously start going through the whole paper, perform data extraction, and at the same time if any irrelevant paper is found we filter it out. In the final phase we collect and read the papers, and extract data for data analysis.

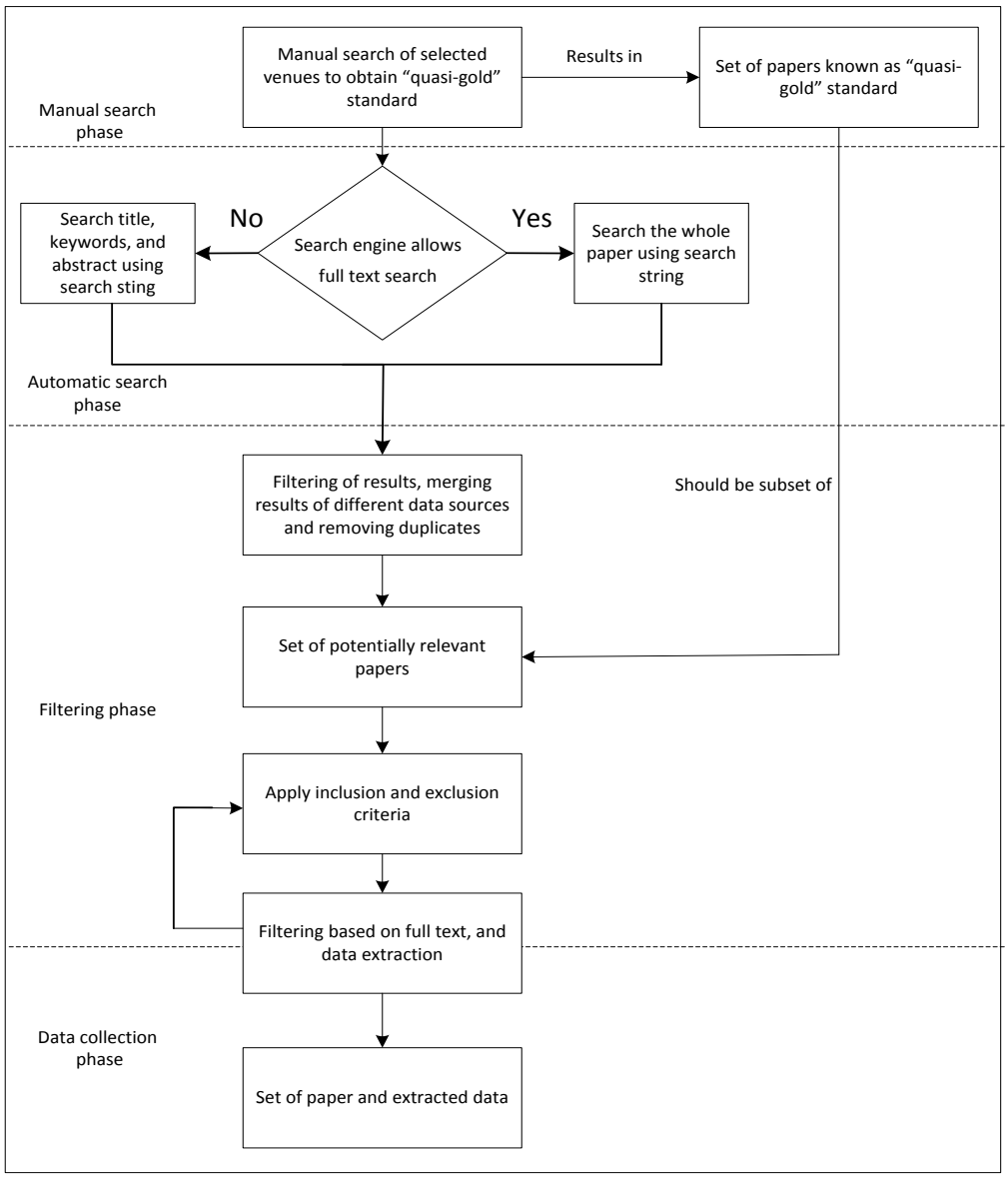

Figure 1- Search process

2.2.5. Inclusion and exclusion criteria

Inclusion criteria

To be selected, a paper needed to cover all the following inclusion criteria:

1. The study should be in the domain of self-adaptive systems.
2. The method presented to manage systems adaptability should be architecture-based. This implies that the study should provide architectural solutions (e.g., architectural models) to handle and reason about the dynamic behavior of the system. In other words, it should be possible to map components of systems 'architectural logic (i.e., self-architectural model) to MAPE-k functionalities.
3. The study should tackle multiple quality requirements as a consequence of applying a self-adaptation method.

Exclusion criteria

A paper was excluded if it fulfilled one of the following exclusion criterions:

1. Study is editorial, position paper, abstract, keynote, opinion, tutorial summary, panel discussion, or technical report. A paper that is not a peer-reviewed scientific paper may not be of acceptable quality or may not provide reasonable amount of information.
2. The study in not in English language.

3. Data Extraction and Synthesis

To execute the data collection and create the classification framework, we stablish a primary scheme for uncertainty dimensions and sources of uncertainty based on (Walker et al., 2003), (Esfahani & Malek, 2013), (Ramirez et al., 2012), (Perez-palacin et al., 2014), and (Refsgaard et al., 2007). The scheme contains most significant aspects of uncertainty which are under investigation in the existing research, and definitions for these aspects (see Table 2, Table 3).

Table 2 - Primary scheme for dimension of uncertainty.

Uncertainty Dimension	Dimension Descriptions
Location (Walker et al., 2003)	"It is an identification of where uncertainty manifests itself within the whole model complex."
Nature (Walker et al., 2003)	"Specifies whether the uncertainty is due to the imperfection of our knowledge, or is due to the inherent variability of the phenomena being described."
Level /Spectrum (Walker et al., 2003),(Esfahani & Malek, 2013)	"Indicates where the uncertainty manifests itself along the spectrum between deterministic knowledge and total ignorance."
Sources (Esfahani & Malek, 2013)	"Factors challenge the confidence with which the adaptation decisions are made." Refers to a variety of uncertainties originating from system models, adaptation actions, systems goals, and executing environment

Table 3 - Sources of uncertainty initial classification schema.

Uncertainty Source	Dimension Descriptions
Model	Refers to a variety of uncertainties originating from system models.
Goals	Refers to a variety of uncertainties originating from system's goal related complications.
Environment	Refers to a variety of uncertainties originating from environments circumstances.

Both Table 2 and Table 3 will be expanded later on by reading the selected primary studies. To complete this classification framework we plan to: 1) identify and list potential locations for uncertainty which are discussed in literature, 2) identify and list different types of uncertainty (i.e., nature), 3) determine common levels in which the uncertainty is addressed in domain of self-adaptive systems, 4) identify and list in which specific times uncertainty is mostly handled, and finally 5) to explore and list all the existing sources of uncertainty, and their definitions. We also aim at categorizing the sources of uncertainty in such a way that they can be mapped into the locations; and moreover, we will present examples for each of the sources form the literature.

Data extraction from the selected primary studies is simultaneously performed as we improve and expand the uncertainty scheme. Therefore, at the end of data extraction phase, we will have our uncertainty classification framework, and an excel file containing detailed mapping of selected primary studies to the uncertainty classification framework. Due to the fact that the classification is still incomplete and under construction during this phase, the mapping of data to the classification framework can be furthered revised and improved during the data analysis phase.

Table 4 lists the data fields we used to extract useful data from the primary studies in order to answer our research questions (RQ). Descriptions of the data fields are provided in tables 1, and 2.

Table 4 - Data form used for data extractions.

Item ID	Data field	Purpose
F1	Author(s) name	RQ1
F2	Title	RQ1
F3	Publication year	RQ1
F4	Venue	RQ1
F5	Location	RQ2
F6	Nature	RQ2
F7	Level/Spectrum	RQ2
F8	Emerging Time	RQ2
F9	Sources	RQ3

4. Data analysis and report

Data extracted from the collected studies will be recoded in excel files to be analyzed and synthesized, and the proposed classification framework will be further improved to answer the research questions. We plan to perform descriptive analysis to synthetize our data. Descriptive analysis is used to present quantitative descriptions for the extracted data, and to summarize the data in a comprehensible and manageable format. If applicable, we will use appropriate statistical methods to reduce large amounts of data to a simpler summary. Combined with plot representations of the analyzed data, data can be used for comparisons across all data, and to answer the research questions. Data analysis process includes the following steps:

1. Review of data, discussion, and reaching consensus among researchers (if applicable).
2. Expanding the classification schemas.
3. Mapping of primary study according to the proposed classification framework.
4. Statistical data analysis to conduct useful information and conclusions, if applicable.

In the reporting phase we present the following sections:

1. A classification framework for architecture-based approaches tackling uncertainty in self-adaptive systems with multiple quality requirements.
2. Extensive answers to research questions, and the importance and usefulness of related findings.
3. A classification of current architecture-based approaches tackling uncertainty in self-adaptive systems with multiple quality requirements according to the proposed classification framework.

As addressed by (Kitchenham & Charters, 2007) it is not possible to select definite data synthesizing approaches in the protocol. Thus, as we proceed with the data extraction, recurring data patterns will help to specify best data synthesis approaches.

5. Limitation and risks

In this section we discuss the limitations and risks that may potentially affect the validity of the systematic literature review and represent solutions to mitigate these threads.

5.1. Bias

The pilot search indicated that, it is not always easy to extract relevant information from the primary studies. Therefore, we may have bias and inaccuracy in the extracted data and creation of classification framework. To mitigate this, we plan to have discussion among researchers and to ask experts to judge accuracy of data when the researchers cannot reach a census on certain extracted data occasionally.

5.2. Reference manager tool bugs

As reference manager tool, we use Mendeley desktop version 1.11, and also its web importer to automatically import papers from digital libraries websites into Mendeley. Mendeley works reasonably fine for a free reference manager tool, however we experience some inconsistencies between desktop and online libraries. This can be simply avoided by synchronizing the libraries every now and then. We also encounter a few errors while importing the results through web importer. We could not figure out the real reason for having the errors, but we assume it could be as a result of performing maintenance on the software. These sorts of errors are normally disappear by themselves and do not need our action.

5.3. Search engine problems

The search engines we use to perform the automatic search in this SLR are considered to be user friendly and robust. However, as we seen so far in the pilot search, some of them, such as IEEE Xplore, are noticeably more powerful and convenient to work with comparing to others (e.g., SpringerLink). First problem we encountered so far is the restricted number of papers citation we can download from certain (e.g., SpringerLink) libraries at once. We solve the first problem by splitting the search into limited publication years or publication titles. This helps to reduce the number of returned results per search and speeds up the downloading process. Second problem is limited filtering options of some of the search engines which results in getting huge number of papers, including many irrelevant papers. This is especially tricky when the library does not allow downloading the papers at once as well. To avoid the repetitive and tedious work of downloading a big number of papers by hand, we filter out (either automatically based on available filtering options or manually based on titles) papers in the digital library and then download the papers into our reference manager tool. This can significantly reduce the volume of downloaded papers, and time spent on downloading citations.

5.4. Domain of Study

One of the main risks of performing an SLR in the domain of self-adaptive systems is lack of a common terminology. This problem emanates from the fact that research in this field is still in an exploratory phase. The lack of consensus on the key terms in the field implies that in the searching phase, we may not cover all the relevant studies on architecture-based self-adaptation (Weyns & Ahmad, 2013). To mitigate the risk, we use a generic search string containing all the mostly used terms, and we avoid a much narrowed search string to prevent missing papers in the automatic search. In addition, we establish “quasi-gold” standard to investigate the trustworthiness of the created search string. Furthermore, we also have a look at the references of the selected primary studies to figure out if we have missed any well-known paper due to the fact that they are out of the search scope. If applicable (i.e., if they match the search scope), we will include them in our final set of selected primary studies.

Bibliography

Ali, M. S., Ali Babar, M., Chen, L., & Stol, K.-J. (2010). A systematic review of comparative evidence of aspect-oriented programming. *Information and Software Technology*, *52*(9), 871–887. Retrieved from http://www.sciencedirect.com/science/article/pii/S0950584910000819

Calinescu, R. (2013). Emerging techniques for the engineering of self-adaptive high-integrity software. *Assurances for Self-Adaptive Systems*, 297–310. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-36249-1_11

Calinescu, R., Member, S., Grunske, L., Kwiatkowska, M., Mirandola, R., & Tamburrelli, G. (2011). Dynamic QoS Management and Optimization in Service-Based Systems, *37*(3), 387–409.

Cheng, S.-W., Garlan, D., & Schmerl, B. (2006). Architecture-based Self-adaptation in the Presence of Multiple Objectives. In *Proceedings of the 2006 International Workshop on Self-adaptation and Self-managing Systems* (pp. 2–8). ACM. doi:10.1145/1137677.1137679

Computing, A., Paper, W., & Edition, T. (2005). An architectural blueprint for autonomic computing ., (June).

Esfahani, N., & Malek, S. (2013). Uncertainty in Self-Adaptive Software Systems. In R. de Lemos, H. Giese, H. A. Müller, & M. Shaw (Eds.), *Software Engineering for Self-Adaptive Systems II* (pp. 214–238). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-35813-5_9

Garlan, D., Schmerl, B., & Steenkiste, P. (2004). Rainbow: architecture-based self-adaptation with reusable infrastructure. In *International Conference on Autonomic Computing, 2004. Proceedings.* (pp. 276–277). IEEE. doi:10.1109/ICAC.2004.1301377

ISO/IEC 25012:2008 - Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Data quality model. (n.d.). Retrieved May 29, 2014, from http://www.iso.org/iso/catalogue_detail.htm?csnumber=35736

ISO/IEC 9126-1 Software Eng. -Product quality - Part 1: Qualitymodel, Int. Standard Organization. (2001).

ISO9126 - Software Quality Characteristics. (n.d.). Retrieved May 29, 2014, from http://www.sqa.net/iso9126.html

Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Retrieved from http://www.citeulike.org/group/14013/article/7874938

Mahdavi-Hezavehi, S., Galster, M., & Avgeriou, P. (2013). Variability in quality attributes of service-based software systems: A systematic literature review. *Information and Software Technology*, *55*(2), 320–343. doi:http://dx.doi.org/10.1016/j.infsof.2012.08.010

Oreizy, P., Medvidovic, N., & Taylor, R. N. (1998). Architecture-based runtime software evolution. *Proceedings of the 20th International Conference on Software Engineering*, *1998*, 177–186. doi:10.1109/ICSE.1998.671114

Perez-palacin, D., Milano, P., & Elettronica, D. (2014). Uncertainties in the Modeling of Self-adaptive Systems : a Taxonomy and an Example of Availability Evaluation, 3–14.

Ramirez, A. J., Jensen, A. C., & Cheng, B. H. C. (2012). A taxonomy of uncertainty for dynamically adaptive systems. *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*. doi:10.1109/SEAMS.2012.6224396

Refsgaard, J. C., van der Sluijs, J. P., Højberg, A. L., & Vanrolleghem, P. a. (2007). Uncertainty in the environmental modelling process – A framework and guidance. *Environmental Modelling & Software*, *22*(11), 1543–1556. doi:10.1016/j.envsoft.2007.02.004

Rotmans, J., Sluijs, J. P. V. A. N. D. E. R., Asselt, M. B. A. V. A. N., Janssen, P., & Krauss, M. P. K. V. O. N. (2003). A Conceptual Basis for Uncertainty Management, *4*(1), 5–17.

Salehie, M., & Tahvildari, L. (2009). Self-adaptive Software: Landscape and Research Challenges. *ACM Trans. Auton. Adapt. Syst.*, *4*(2), 14:1–14:42. doi:10.1145/1516533.1516538

Weyns, D., & Ahmad, T. (2013). Claims and evidence for architecture-based self-adaptation: a systematic literature review. *Software Architecture*. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-39031-9_22

Zhang, H., & Ali Babar, M. (2010). On searching relevant studies in software engineering. British Informatics Society Ltd. Retrieved from http://ulir.ul.ie/handle/10344/730